Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 9365340, 29 pages
https://doi.org/10.1155/2020/9365340

Research Article

Deep Learning Based Abstractive Text Summarization: Approaches, Datasets, Evaluation Measures, and Challenges

Dima Suleiman and Arafat Awajan

Princess Sumaya University for Technology, Amman, Jordan

Correspondence should be addressed to Dima Suleiman; d.suleiman@psut.edu.jo

Received 24 April 2020; Revised 1 July 2020; Accepted 25 July 2020; Published 24 August 2020

Academic Editor: Dimitris Mourtzis

Copyright © 2020 Dima Suleiman and Arafat Awajan. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In recent years, the volume of textual data has rapidly increased, which has generated a valuable resource for extracting and analysing information. To retrieve useful knowledge within a reasonable time period, this information must be summarised. This paper reviews recent approaches for abstractive text summarisation using deep learning models. In addition, existing datasets for training and validating these approaches are reviewed, and their features and limitations are presented. The Gigaword dataset is commonly employed for single-sentence summary approaches, while the Cable News Network (CNN)/Daily Mail dataset is commonly employed for multisentence summary approaches. Furthermore, the measures that are utilised to evaluate the quality of summarisation are investigated, and Recall-Oriented Understudy for Gisting Evaluation 1 (ROUGE1), ROUGE2, and ROUGE-L are determined to be the most commonly applied metrics. The challenges that are encountered during the summarisation process and the solutions proposed in each approach are analysed. The analysis of the several approaches shows that recurrent neural networks with an attention mechanism and long short-term memory (LSTM) are the most prevalent techniques for abstractive text summarisation. The experimental results show that text summarisation with a pretrained encoder model achieved the highest values for ROUGE1, ROUGE2, and ROUGE-L (43.85, 20.34, and 39.9, respectively). Furthermore, it was determined that most abstractive text summarisation models faced challenges such as the unavailability of a golden token at testing time, out-of-vocabulary (OOV) words, summary sentence repetition, inaccurate sentences, and fake facts.

1. Introduction

Currently, there are vast quantities of textual data available, including online documents, articles, news, and reviews that contain long strings of text that need to be summarised [1]. The importance of text summarisation is due to several reasons, including the retrieval of significant information from a long text within a short period, easy and rapid loading of the most important information, and resolution of the problems associated with the criteria needed for summary evaluation [2]. Due to the evolution and growth of automatic text summarisation methods, which have provided significant results in many languages, these methods need to be reviewed and summarised. Therefore, in this review, we surveyed the most recent methods and focused on the techniques, datasets, evaluation measures, and challenges of each approach, in addition to the manner in which each method addressed challenges.

Applications such as search engines and news websites use text summarisation [1]. In search engines, previews are produced as snippets, and news websites generate headlines to describe the news to facilitate knowledge retrieval [3, 4]. Text summarisation can be divided into several categories based on function, genre, summary context, type of summarizer, and number of documents [5]; one specific text summarisation classification approach divides the summarisation process into extractive and abstractive categories [6].

Extractive summarisation extracts or copies some parts from the original text based on scores computed using either statistical features or linguistic features, while abstractive summarisation rephrases the original text to generate new phrases that may not be in the original text, which is

considered a difficult task for a computer. As abstractive text summarisation requires an understanding of the document to generate the summary, advanced machine learning techniques and extensive natural language processing (NLP) are required. Thus, abstractive summarisation is harder than extractive summarisation since abstractive summarisation requires real-word knowledge and semantic class analysis [7]. However, abstractive summarisation is also better than extractive summarisation since the summary is an approximate representation of a human-generated summary, which makes it more meaningful [8]. For both types, acceptable summarisation should have the following: sentences that maintain the order of the main ideas and concepts presented in the original text, minimal to no repetition, sentences that are consistent and coherent, and the ability to remember the meaning of the text, even for long sentences [7]. In addition, the generated summary must be compact while conveying important information about the original text [2, 9].

Abstractive text summarisation approaches include structured and semantic-based approaches. Structured approaches encode the crucial features of documents using several types of schemas, including tree, ontology, lead and body phrases, and template and rule-based schemas, while semantic-based approaches are more concerned with the semantics of the text and thus rely on the information representation of the document to summarise the text. Semantic-based approaches include the multimodal semantic method, information item method, and semantic graph-based method [10–17].

Deep learning techniques were employed in abstractive text summarisation for the first time in 2015 [18], and the proposed model was based on the encoder-decoder architecture. For these applications, deep learning techniques have provided excellent results and have been extensively employed in recent years.

Raphal et al. surveyed several abstractive text summarisation processes in general [19]. Their study differentiated between different model architectures, such as reinforcement learning (RL), supervised learning, and attention mechanism. In addition, comparisons in terms of word embedding, data processing, training, and validation had been performed. However, there are no comparisons of the quality of several models that generated summaries.

Furthermore, both extractive and abstractive summarisation models were summarised in [20, 21]. In [20], the classification of summarisation tasks was based on three factors: input factors, purpose factors, and output factors. Dong and Mahajani et al. surveyed only five abstractive summarisation models each. On the other hand, Mahajani et al. focused on the datasets and training techniques in addition to the architecture of several abstractive summarisation models [21]. However, the quality of the generated summary of the different techniques and the evaluation measures were not discussed.

Shi et al. presented a comprehensive survey of several abstractive text summarisation models, which are based on sequence-to-sequence encoder-decoder architecture for convolutional and RNN seq2seq models. The focus was the structure of the network, training strategy, and the algorithms employed to generate the summary [22]. Although several papers have analysed abstractive summarisation models, few papers have performed a comprehensive study [23]. Moreover, most of the previous surveys covered the techniques until 2018, even though surveys were published in 2019 and 2020, such as [20, 21]. In this review, we addressed most of the recent deep learning-based RNN abstractive text summarisation models. Furthermore, this survey is the first to address recent techniques applied in abstractive summarisation, such as Transformer.

This paper provides an overview of the approaches, datasets, evaluation measures, and challenges of deep learning-based abstractive text summarisation, and each topic is discussed and analysed. We classified the approaches based on the output type into: single-sentence summary and multi-sentence summary approaches. Also, within each classification, we compared between the approaches in terms of architecture, dataset, dataset preprocessing, evaluation, and results. The remainder of this paper is organised as follows: Section 2 introduces a background of several deep learning models and techniques, such as the recurrent neural network (RNN), bidirectional RNN, attention mechanisms, long short-term memory (LSTM), gated recurrent unit (GRU), and sequence-to-sequence models. Section 3 describes the most recent single-sentence summarisation approaches, while the multisentence summarisation approaches are covered in Section 4. Section 5 and Section 6 investigate datasets and evaluation measures, respectively. Section 7 discusses the challenges of the summarisation process and solutions to these challenges. Conclusions and discussion are provided in Section 8.

2. Background

Deep learning analyses complex problems to facilitate the decision-making process. Deep learning attempts to imitate what the human brain can achieve by extracting features at different levels of abstraction. Typically, higher-level layers have fewer details than lower-level layers [24]. The output layer will produce an output by nonlinearly transforming the input from the input layer. The hierarchical structure of deep learning can support learning. The level of abstraction of a certain layer will determine the level of abstraction of the next layer since the output of one layer will be the input of the next layer. In addition, the number of layers determines the deepness, which affects the level of learning [25].

Deep learning is applied in several NLP tasks since it facilitates the learning of multilevel hierarchal representations of data using several data processing layers of nonlinear units [24, 26–28]. Various deep learning models have been employed for abstractive summarisation, including RNNs, convolutional neural networks (CNNs), and sequence-to-sequence models. We will cover deep learning models in more detail in this section.

2.1. RNN Encoder-Decoder Summarization. RNN encoder-decoder architecture is based on the sequence-to-sequence model. The sequence-to-sequence model maps the input sequence in the neural network to a similar sequence that consists of characters, words, or phrases. This model is

utilised in several NLP applications, such as machine translation and text summarisation. In text summarisation, the input sequence is the document that needs to be summarised, and the output is the summary [29, 30], as shown in Figure 1.

An RNN is a deep learning model that is applied to process data in sequential order such that the input of a certain state depends on the output of the previous state [31, 32]. For example, in a sentence, the meaning of a word is closely related to the meaning of the previous words. An RNN consists of a set of hidden states that are learned by the neural network. An RNN may consist of several layers of hidden states, where states and layers learn different features. The last state of each layer represents the whole inputs of the layer since it accumulates the values of all previous states [5]. For example, the first layer and its state can be employed for part-of-speech tagging, while the second layer learns to create phrases. In text summarisation, the input for the RNN is the embedding of words, phrases, or sentences, and the output is the word embedding of the summary [5].

In the RNN encoder-decoder model, at the encoder side, at certain hidden states, the vector representation of the current input word and the output of the hidden states of all previous words are combined and fed to the next hidden state. As shown in Figure 1, the vector representation of the word W3 and the output of the hidden states he1 and he2 are combined and fed as input to the hidden states he3. After feeding all the words of the input string, the output generated from the last hidden state of the encoder are fed to the decoder as a vector referred to as the context vector [29]. In addition to the context vector, which is fed to the first hidden state of the decoder, the start-of-sequence symbol ⟨SOS⟩ is fed to generate the first word of the summary from the headline (assume W5, as shown in Figure 1). In this case, W5 is fed as the input to the next decoder hidden state. Each generated word is passed as an input to the next decoder hidden state to generate the next word of the summary. The last generated word is the end-of-sequence symbol ⟨EOS⟩. Before generating the summary, each output from the decoder will take the form of a distributed representation before it is sent to the softmax layer and attention mechanism to generate the next summary [29].

2.2. Bidirectional RNN. Bidirectional RNN consists of forward RNNs and backward RNNs. Forward RNNs generate a sequence of hidden states after reading the input sequence from left to right. On the other hand, the backward RNNs generate a sequence of hidden states after reading the input sequence from right to left. The representation of the input sequence is the concatenation of the forward and backward RNNs [33]. Therefore, the representation of each word depends on the representation of the preceding (past) and following (future) words. In this case, the context will contain the words to the left and the words to the right of the current word [34].

Using bidirectional RNN enhances the performance. For example, if we have the following input text "Sara ate a delicious pizza at dinner tonight," in this case, assume that we want to predict the representation of the word "dinner," using bidirectional RNN and the forward LSTMs represent "Sara ate a delicious pizza at" while the backward LSTM represents "tonight." Considering the word "tonight" when representing the word "dinner" provides better results.

On the other hand, using the bidirectional RNN at the decoder size minimizes the probability of the wrong prediction. The reason for this is that the unidirectional RNN only considers the previous prediction and reason only about the past. Therefore, if there is an error in previous prediction, the error will accumulate in all subsequent predictions, and this problem can be addressed using the bidirectional RNN [35].

2.3. Gated Recurrent Neural Networks (LSTM and GRU). Gated RNNs are employed to solve the problem of vanishing gradients, which occurs when training a long sequence using an RNN. This problem can be solved by allowing the gradients to backpropagate along a linear path using gates, where each gate has a weight and a bias. Gates can control and modify the amount of information that flows between hidden states. During training, the weights and biases of the gates are updated. The most popular gated RNNs are LSTM [36] and GRU [37], which are two variants of an RNN.

2.3.1. Long Short-Term Memory (LSTM). The repeating unit of the LSTM architecture consists of input/read, memory/update, forget, and output gates [5, 7], but the chaining structure is the same as that of an RNN. The four gates share information with each other; thus, information can flow in loops for a long period of time. The four gates of each LSTM unit, which are shown in Figures 2 and 3, are discussed here.

(1) Input Gate. In the first timestep, the input is a vector that is initialised randomly, while in subsequent steps, the input of the current step is the output (content of the memory cell) of the previous step. In all cases, the input is subject to element-wise multiplication with the output of the forget gate. The multiplication result is added to the current memory gate output.

(2) Forget Gate. A forget gate is a neural network with one layer and a sigmoid activation function. The value of the sigmoid function will determine if the information of the previous state should be forgotten or remembered. If the sigmoid value is 1, then the previous state will be remembered, but if the sigmoid value is 0, then the previous state will be forgotten. In language modelling, for example, the forget gate remembers the gender of the subject to produce the proper pronouns until it finds a new subject. There are four inputs for the forget gate: the output of the previous block, the input vector, the remembered information from the previous block, and the bias.

(3) Memory Gate. The memory gate controls the effect of the remembered information on the new information. The memory gate consists of two neural networks. The first network has the same structure as the forget gate but a different bias, and the second neural network has a tanh activation function and is utilised to generate the new

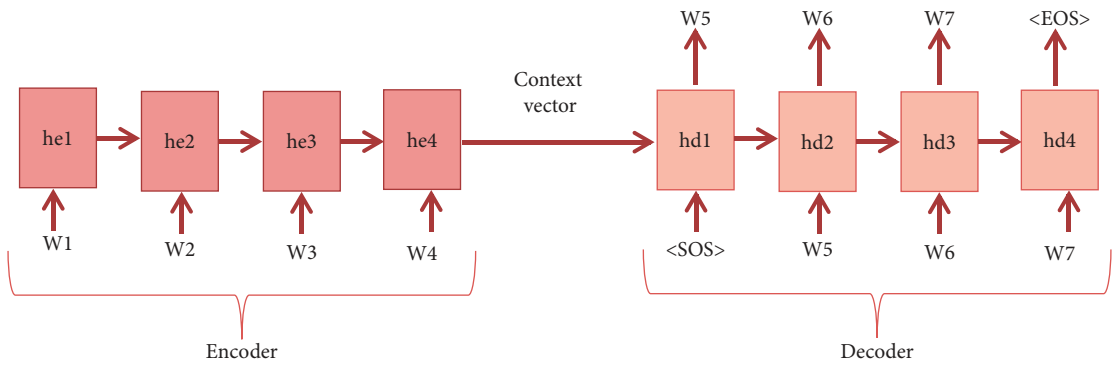

Figure 1: Sequence-to-sequence; the last hidden state of the encoder is fed as input to the decoder with the symbol EOS [51].

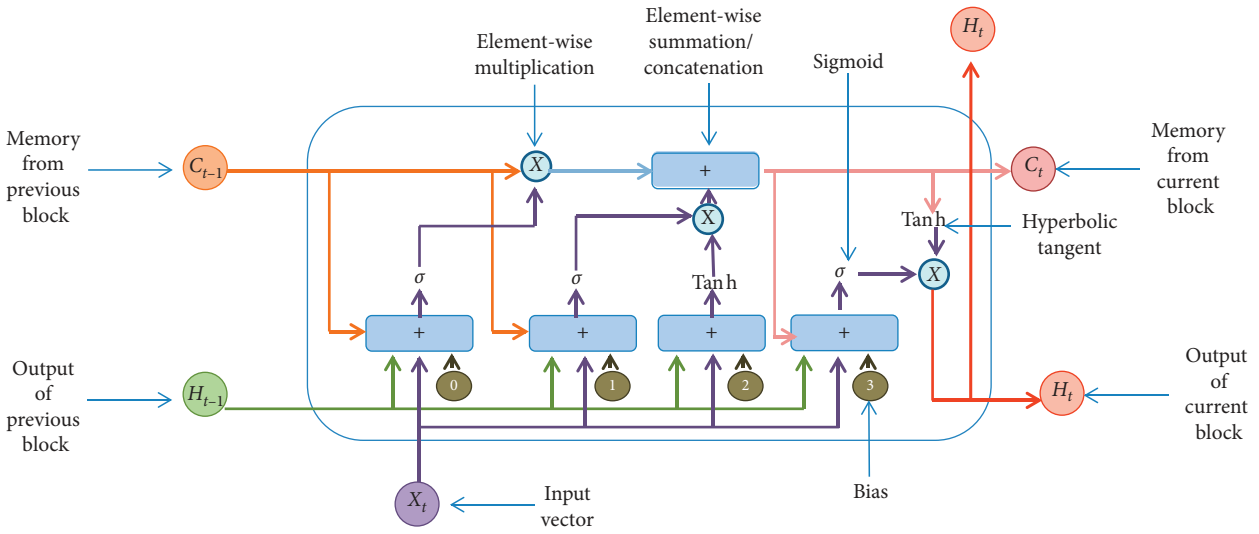

Figure 2: LSTM unit architecture [5].

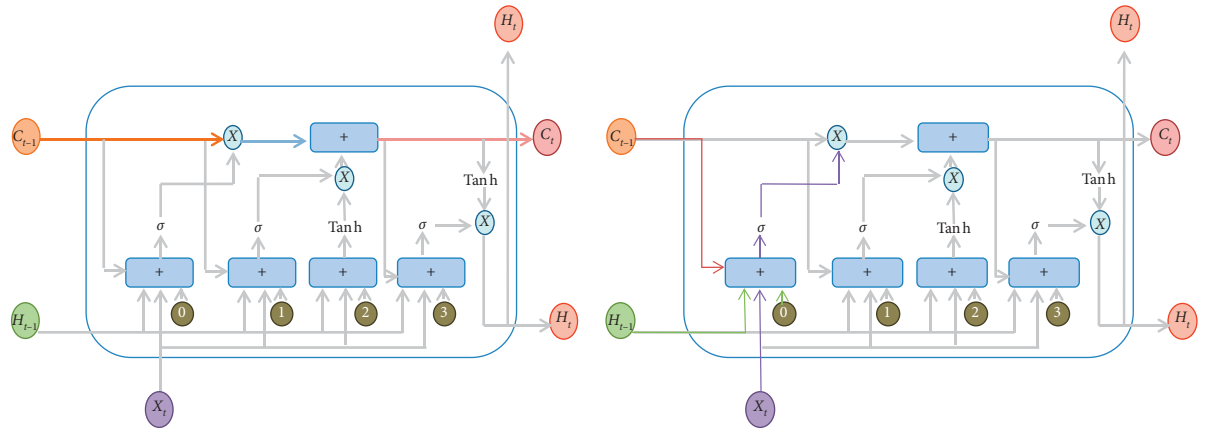

Figure 3: Continued.

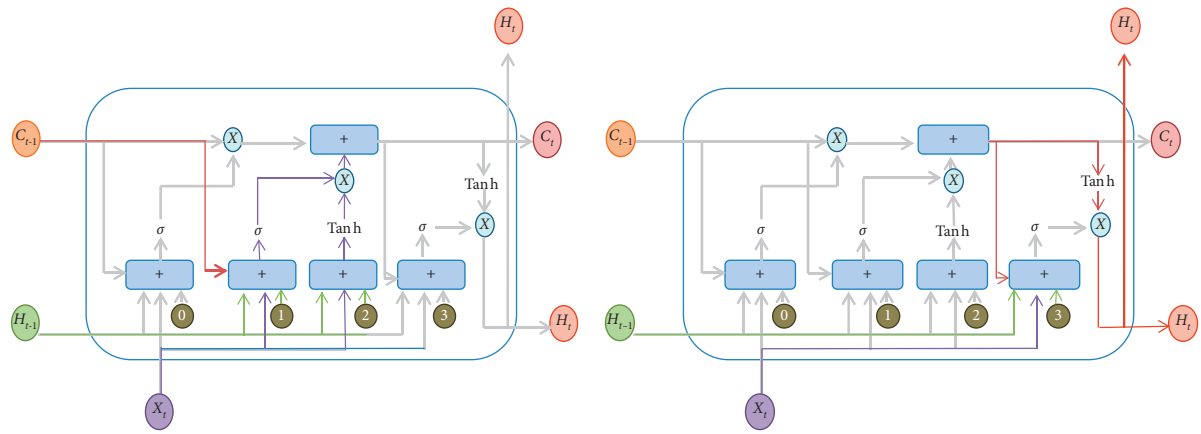

FIGURE 3: LSTM unit gates [5]: (a) input gate; (b) forget gate; (c) memory gate; (d) output gate.

information. The new information is formed by adding the old information to the result of the element-wise multiplication of the output of the two memory gate neural networks.

(4) Output Gate. The output gates control the amount of new information that is forwarded to the next LSTM unit. The output gate is a neural network with a sigmoid activation function that considers the input vector, the previous hidden state, the new information, and the bias as input. The output of the sigmoid function is multiplied by the tanh of the new information to produce the output of the current block.

2.3.2. Gated Recurrent Unit (GRU). A GRU is a simplified LSTM with two gates, a reset gate and an update gate, and there is no explicit memory. The previous hidden state information is forgotten when all the reset gate elements approach zero; then, only the input vector affects the candidate hidden state. In this case, the update gate acts as a forget gate. LSTM and GRU are commonly employed for abstractive summarisation since LSTM has a memory unit that provides extra control; however, the computation time of the GRU is reduced [38]. In addition, while it is easier to tune the parameters with LSTM, the GRU takes less time to train [30].

2.4. Attention Mechanism. The attention mechanism was employed for neural machine translation [33] before being utilised for NLP tasks such as text summarisation [18]. A basic encoder-decoder architecture may fail when given long sentences since the size of encoding is fixed for the input string; thus, it cannot consider all the elements of a long input. To remember the input that has a significant impact on the summary, the attention mechanism was introduced [29]. The attention mechanism is employed at each output word to calculate the weight between the output word and every input word; the weights add to one. The advantage of using weights is to show which input word must receive attention with respect to the output word. The weighted average of the last hidden layers of the decoder in the current step is calculated after passing each input word and fed to the softmax layer along the last hidden layers [39].

2.5. Beam Search. Beam search and greedy search are very similar; however, while greedy search considers only the best hypothesis, beam search considers b hypotheses, where b represents the beam width or beam size [5]. In text summarisation tasks, the decoder utilises the final encoder representation to generate the summary from the target vocabulary. In each step, the output of the decoder is a probability distribution over the target word. Thus, to obtain the output word from the learned probability, several methods can be applied, including (1) greedy sampling, which selects the distribution mode, (2) 1-best or beam search, which selects the best output, and (3) n-best or beam search, which select several outputs. When n-best beam search is employed, the top b most relevant target words are selected from the distribution and fed to the next decoder state. The decoder keeps only the top k1 of k words from the different inputs and discards the rest.

2.6. Distributed Representation (Word Embedding). A word embedding is a word distributional vector representation that represents the syntax and semantic features of words [40]. Words must be converted to vectors to handle various NLP challenges such that the semantic similarity between words can be calculated using cosine similarity, Euclidean distance, etc. [41–43]. In NLP tasks, the word embeddings of the words are fed as inputs to neural network models. In the recurrent neural network encoder-decoder architecture, which is employed to generate the summaries, the input of the model is the word embedding of the text, and the output is the word embedding of the summary.

In NLP, there are several word embedding models, such as Word2Vec, GloVe, FastText, and Bidirectional Encoder Representations from Transformers (BERT), which are the most recently employed word embedding models [41, 44–47]. The Word2Vec model consists of two approaches, skip-gram and continuous bag-of-words (CBOW), which both depend on the context window [41]. On the other hand, GloVe represents the

global vector, which is based on statistics of the global corpus instead of the context window [44]. FastText extends the skip-gram of the Word2Vec model by using the subword internal information to address the out-of-vocabulary (OOV) terms [46]. In FastText, the subword components are composed to build the vector representation of the words, which facilitates representation of the word morphology and lexical similarity. The BERT word embedding model is based on a multilayer bidirectional transformer encoder [47, 48]. Instead of using sequential recurrence, the transformer neural network utilises parallel attention layers. BERT creates a single large transformer by combining the representations of the words and sentences. Furthermore, BERT is pretrained with an unsupervised objective over a large amount of text.

2.7. Transformers. The contextual representations of language are learned from large corpora. One of the new language representations, which extend word embedding models, is referred to as BERT mentioned in the previous section [48]. In BERT, two tokens are inserted to the text. The first token (CLS) is employed to aggregate the whole text sequence information. The second token is (SEP); this token is inserted at the end of each sentence to represent it. The resultant text consists of tokens, where each token is assigned three types of embeddings: token, segmentation, and position embeddings. Token embedding is applied to indicate the meaning of a token. Segmentation embedding identifies the sentences, and position embedding determines the position of the token. The sum of the three embeddings is fed to the bidirectional transformer as a single vector. Pretrained word embedding vectors are more precise and rich with semantic features. BERT has the advantage of fine-tuning (based on the objectives of certain tasks) and feature-based methods. Moreover, transformers compute the presentation of the input and output by using self-attention, where the self-attention enables the learning of the relevance between the "word-pair" [47].

3. Single-Sentence Summary

Recently, the RNN has been employed for abstractive text summarisation and has provided significant results. Therefore, we focus on abstractive text summarisation based on deep learning techniques, especially the RNN [49]. We discussed the approaches that have applied deep learning for abstractive text summarisation since 2015. RNN with an attention mechanism was mostly utilised for abstractive text summarisation. We classified the research according to summary type (i.e., single-sentence or multisentence summary), as shown in Figure 4. We also compared the approaches in terms of encoder-decoder architecture, word embedding, dataset and dataset preprocessing, and evaluations and results. This section covers single-sentence summary methods, while Section 4 covers multisentence summary methods. Single-sentence summary methods include a neural attention model for abstractive sentence summarisation [18], abstractive sentence summarisation with attentive RNN (RAS) [39], quasi-RNN [50], a method for generating news headlines with RNNs [29], abstractive text summarisation using an attentive sequence-to-sequence RNN [38], neural text summarisation [51], selective encoding for abstractive sentence summarisation (SEASS) [52], faithful to the original: fact aware neural abstractive summarization (FTSumg) [53], and the improving transformer with sequential context [54].

3.1. Abstractive Summarization Architecture

3.1.1. Feedforward Architecture. Neural networks were first employed for abstractive text summarisation by Rush et al. in 2015, where a local attention-based model was utilised to generate summary words by conditioning it to input sentences [18]. Three types of encoders were applied: the bag-of-words encoder, the convolution encoder, and the attention-based encoder. The bag-of-words model of the embedded input was used to distinguish between stop words and content words; however, this model had a limited ability to represent continuous phrases. Thus, a model that utilised the deep convolutional encoder was employed to allow the words to interact locally without the need for context. The convolutional encoder model can alternate between temporal convolution and max-pooling layers using the standard time-delay neural network (TDNN) architecture; however, it is limited to a single output representation. The limitation of the convolutional encoder model was overcome by the attention-based encoder. The attention-based encoder was utilised to exploit the learned soft alignment to weight the input based on the context to construct a representation of the output. Furthermore, the beam-search decoder was applied to limit the number of hypotheses in the summary.

3.1.2. RNN Encoder-Decoder Architecture

(1) LSTM-RNN. An abstractive sentence summarisation model that employed a conditional recurrent neural network (RNN) to generate the summary from the input is referred to as a recurrent attentive summariser (RAS) [39]. A RAS is an extension of the work in [18]. In [18], the model employed a feedforward neural network, while the RAS employed an RNN-LSTM. The encoder and decoder in both models were trained using sentence-summary pair datasets, but the decoder of the RAS improved the performance since it considered the position information of the input words. Furthermore, previous words and input sentences were employed to produce the next word in the summary during the training phase.

Lopyrev [29] proposed a simplified attention mechanism that was utilised in an encoder-decoder RNN to generate headlines for news articles. The news article was fed into the encoder one word at a time and then passed through the embedding layer to generate the word representation. The experiments were conducted using simple and complex attention mechanisms. In the simple attention mechanism, the last layer after processing the input in the encoding was divided into two parts: one part for calculating the attention weight vector, and one part for calculating the context vector, as shown in Figure 5(a). However, in the complex attention mechanism, the last layer was employed to calculate the attention weight vector and context vector without

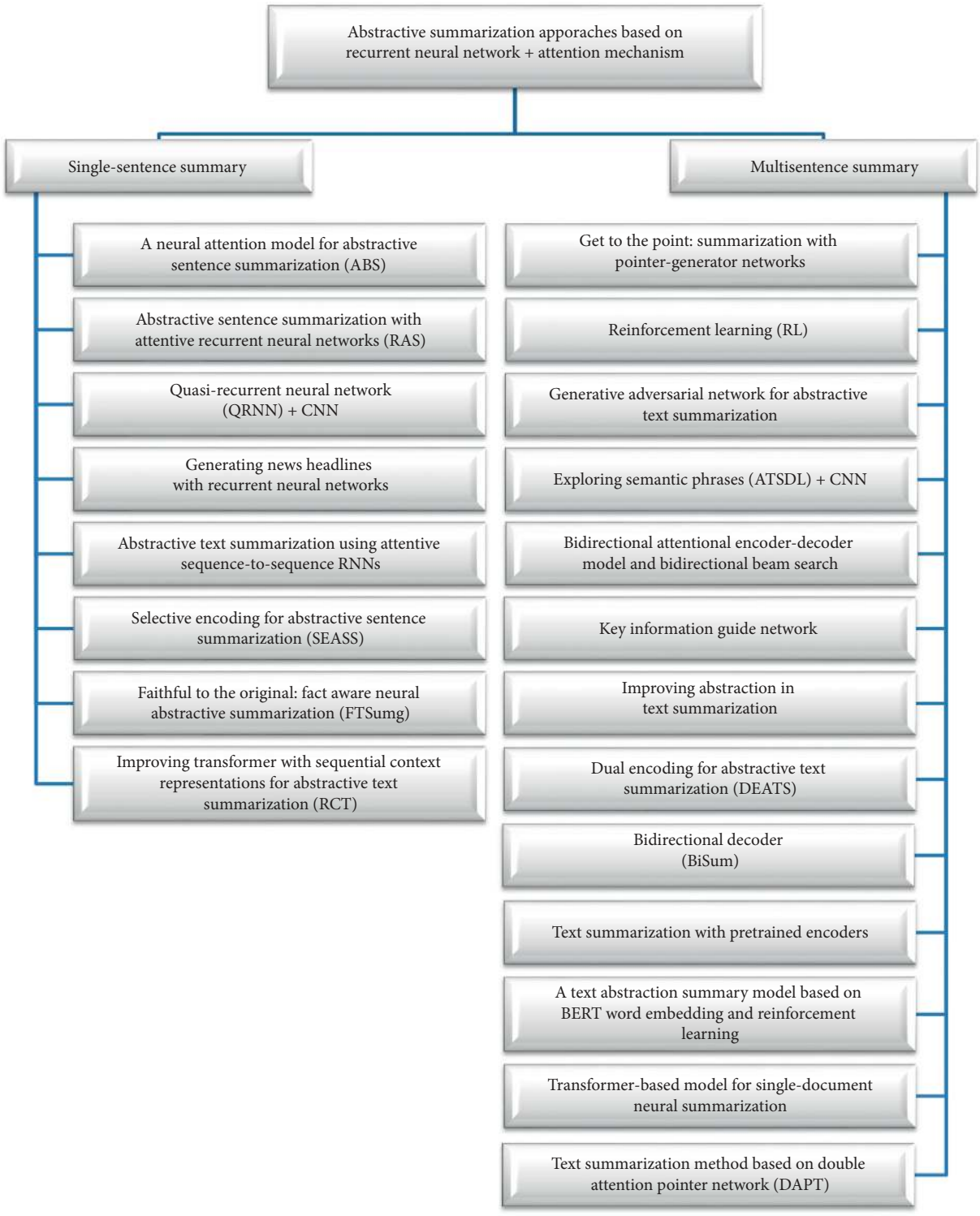

Figure 4: Taxonomy of several approaches that use a recurrent neural network and attention mechanism in abstractive text summarisation based on the summary type.

fragmentation, as shown in Figure 5(b). In both figures, the solid lines indicate the part of the hidden state of the last layer that is employed to compute the context vector, while the dashed lines indicate the part of the hidden state of the last layer that is applied to compute the attention weight vector. The same difference was seen on the decoder side: in the simple attention mechanism, the last layer was divided into two parts (one part was passed to the softmax layer, and the other part was applied to calculate the attention weight), while in the complex attention mechanism, no such division was made. A beam search at the decoder side was performed during testing to extend the sequence of the probability.

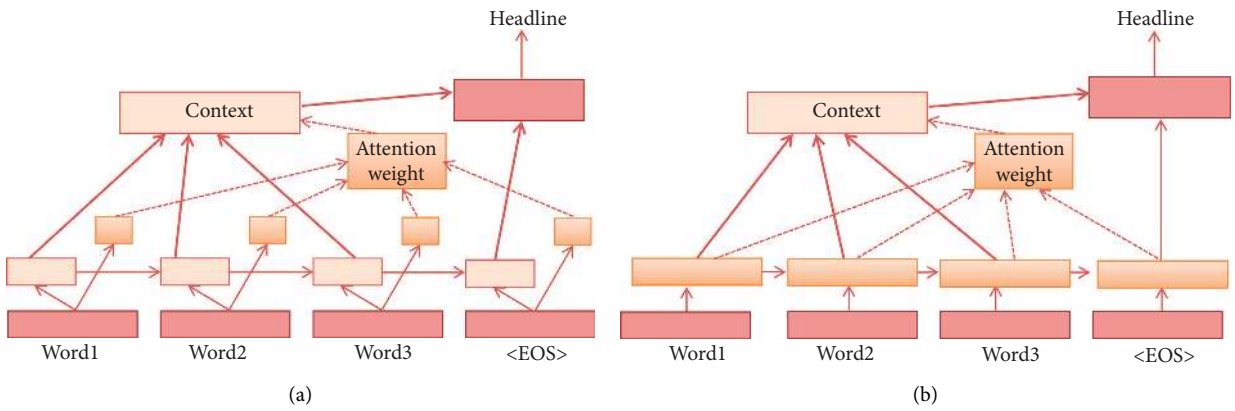

Figure 5: (a) Simple attention and (b) complex attention [29].

The encoder-decoder RNN and sequence-to-sequence models were utilised in [55], which mapped the inputs to the target sequences; the same approach was also employed in [38, 51]. Three different methods for global attention were proposed for calculating the scoring functions, including dot product scoring, the bilinear form, and the scalar value calculated from the projection of the hidden states of the RNN encoder [38]. The model applied LSTM cells instead of GRU cells (both LSTM and GRU are commonly employed for abstractive summarisation tasks since LSTM has a memory unit that provides control but the computation time of GRU is lower). Three models were employed: the first model applied unidirectional LSTM in both the encoder and the decoder; the second model was implemented using bidirectional LSTM in the encoder and unidirectional LSTM in the decoder; and the third model utilised a bidirectional LSTM encoder and an LSTM decoder with global attention. The first hidden state of the decoder is the concatenation of all backward and forward hidden states of the encoder. The use of attention in an encoder-decoder neural network generates a context vector at each timestep. For the local attention mechanism, the context vector is conditioned on a subset of the encoder's hidden states, while for the global attention mechanism, the vector is conditioned on all the encoder's hidden states. After generating the first decoder output, the next decoder input is the word embedding of the output of the previous decoder step. The affine transformation is used to convert the output of the decoder LSTM to a dense vector prediction due to the long training time needed before the number of hidden states is the same as the number of words in the vocabulary.

Khandelwal [51] employed a sequence-to-sequence model that consists of an LSTM encoder and LSTM decoder for abstractive summarisation of small datasets. The decoder generated the output summary after reading the hidden representations generated by the encoder and passing them to the softmax layer. The sequence-to-sequence model does not memorize information, so generalization of the model is not possible. Thus, the proposed model utilised imitation learning to determine whether to choose the golden token (i.e., reference summary token) or the previously generated output at each step.

(2) GRU-RNN. A combination of the elements of the RNN and convolutional neural network (CNN) was employed in an encoder-decoder model that is referred to as a quasi-recurrent neural network (QRNN) [50]. In the QRNN, the GRU was utilised in addition to the attention mechanism. The QRNN was applied to address the limitation of parallelisation, which aimed to obtain the dependencies of the words in previous steps via convolution and "fo-pooling," which were performed in parallel, as shown in Figure 6. The convolution in the QRNN can be either mass convolution (considering previous time-steps only) or centre convolution (considering future time-steps). The encoder-decoder model employed two neural networks: the first network applied the centre convolution of QRNN and consisted of multiple hidden layers that were fed by the vector representation of the words, and the second network comprised neural attention and considered as input the encoder hidden layers to generate one word of a headline. The decoder accepted the previously generated headline word and produced the next word of the headline; this process continued until the headline was completed.

SEASS is an extension of the sequence-to-sequence recurrent neural network that was proposed in [52]. The selective encoding for the abstractive sentence summarisation (SEASS) approach includes a selective encoding model that consists of an encoder for sentences, a selective gate network, and a decoder with an attention mechanism, as shown in Figure 7. The encoder uses a bidirectional GRU, while the decoder uses a unidirectional GRU with an attention mechanism. The encoder reads the input words and their representations. The meaning of the sentences is applied by the selective gate to choose the word representations for generating the word representations of the sentence. To produce an excellent summary and accelerate the decoding process, a beam search was selected as the decoder.

On the other hand, dual attention was applied in [53]. The proposed dual attention approach consists of three modules: two bidirectional GRU encoders and one dual attention decoder. The decoder has a gate network for context selection, as shown in Figure 8, and employs copying and coverage mechanisms. The outputs of the encoders are two context vectors: one context vector for sentences and one context vector for the relation, where the relation may be

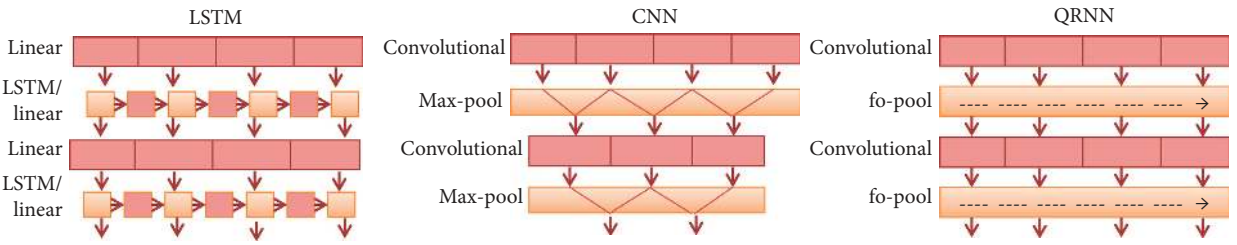

Figure 6: Comparison of the CNN, LSTM, and QRNN models [50].

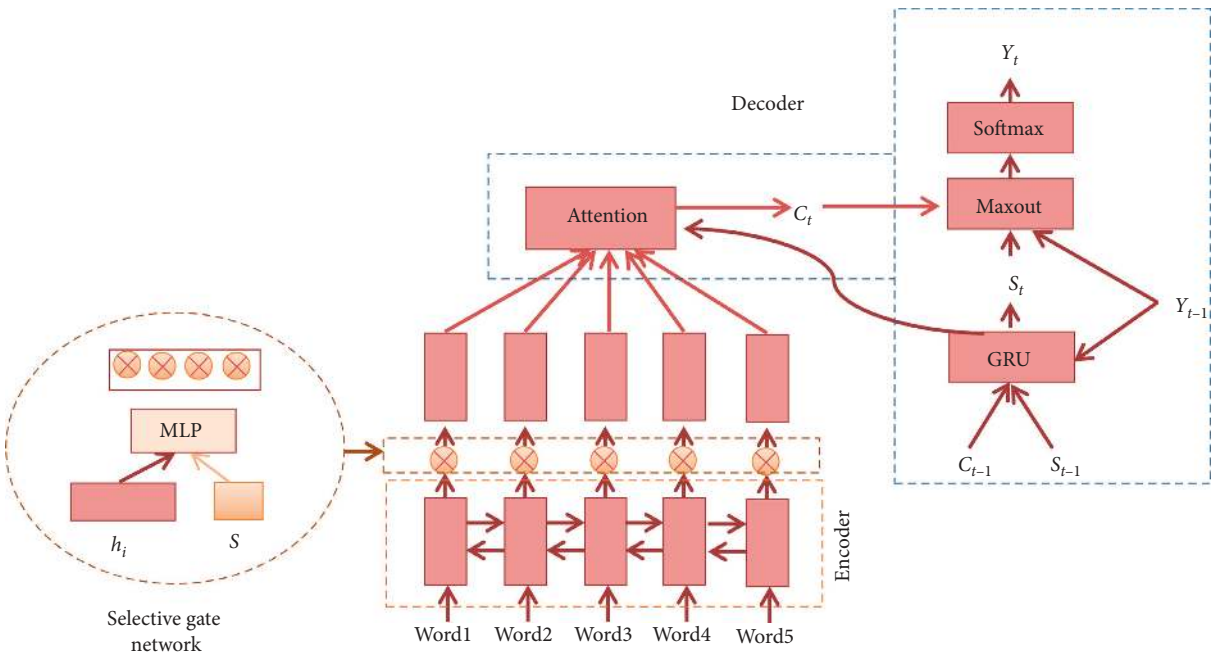

Figure 7: Selective encoding for abstractive sentence summarisation (SEASS) [52].

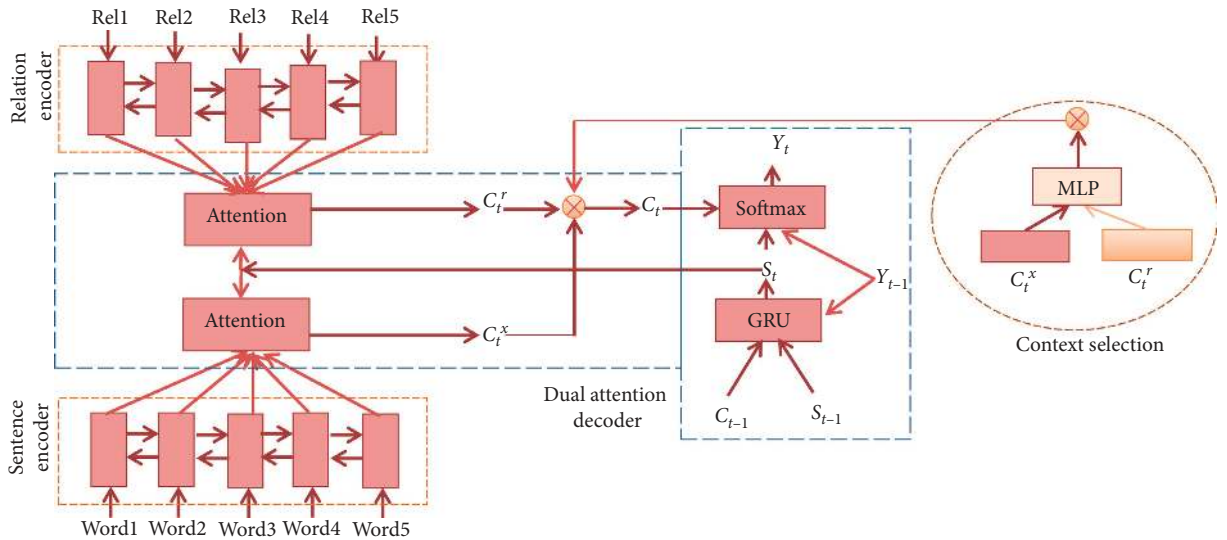

Figure 8: Faithful to the original [53].

a triple or tuple relation. A triple relation consists of the subject, predicate, and object, while the tuple relation consists of either (subject and predicate) or (predicate and subject). Sometimes the triple relation cannot be extracted; in this case, two tuple relations are utilised. The decoder gate merges both context vectors based on their relative association.

(3) Others. The long-sequence poor semantic representation of abstractive text summarisation approaches, which are based on an RNN encoder-decoder framework, was addressed using the RC-Transformer (RCT) [54]. An RCT is an RNN-based abstractive text summarisation model that is composed of two encoders (RC encoder and transformer encoder) and one decoder. The transformer shows an advantage in parallel computing in addition to retrieving the global context semantic relationships. On the other hand, sequential context representation was achieved by a second encoder of the RCT-Transformer. Word ordering is very crucial for abstractive text summarisation, which cannot be obtained by positioning encoding. Therefore, an RCT utilised two encoders to address the problem of a shortage of sequential information at the word level. A beam search was utilised at the decoder. Furthermore, Cai et al. compared the speed of the RCT model and that of the RNN-based model and concluded that the RCT is 1.4x and 1.2x faster.

3.2. Word Embedding. In the QRNN model, GloVe word embedding, which was pretrained using the Wikipedia and Gigaword datasets, was performed to represent the text and summary [50]. In the first model, the proposed model by Jobson et al., the word embedding, randomly initialised and updated during training, while GloVe word embedding was employed to represent the words in the second and third models [38]. In a study by Cai et al., Transformer was utilised [54].

3.3. Dataset and Dataset Preprocessing. In the model that was proposed by Rush et al., datasets were preprocessed via PTB tokenization by using "#" to replace all digits, conversion of all letters to lowercase letters, and the use of "UNK" to replace words that occurred fewer than 5 times [18]. The model was trained with any input-output pairs due to the shortage of constraints for generating the output. The training process was carried out on the Gigaword datasets, while the summarisation evaluation was conducted on DUC2003 and DUC2004 [18]. Furthermore, the proposed model by Chopra et al. was trained using the Gigaword corpus with sentence separation and tokenisation [39]. To form sentence-summary pairs, each headline of the article was paired with the first sentence of the article. The same preprocessing steps of the data in [18] were performed in [39]. Moreover, the Chopra et al. model was evaluated using the DUC2004 dataset, which consists of 500 pairs.

Gigaword datasets were also employed by the QRNN model [50]. Furthermore, articles that started with sentences that contained more than 50 words or headlines with more than 25 words were removed. Moreover, the words in the articles and their headlines were converted to lowercase words, and the data points were split into short, medium, and long sentences, based on the lengths of the sentences, to avoid extra padding.

Lopyrev and Jobson et al. trained the model using Gigaword after processing the data. In the Lopyrev model, the most crucial preprocessing steps for both the text and the headline were tokenisation and character conversion to lowercase [29]. In addition, only the characters of the first paragraph were retained, and the length of the headline was fixed between 25 and 50 words. Moreover, the no-headline articles were disregarded, and the ⟨unk⟩ symbol was used to replace rare words.

Khandelwal employed the Association for Computational Linguistics (ACL) Anthology Reference Corpus, which consists of 16,845 examples for training and 500 examples for testing, and they were considered small datasets, in experiments [51]. The abstract included the first three sentences, and the unigram that overlaps between the title and the abstract was also calculated. There were 25 tokens in the summary, and there were a maximum of 250 tokens in the input text.

The English Gigaword dataset, DUC2004 corpus, and MSR-ATC were selected to train and test the SEASS model [52]. Moreover, the experiments of the Cao et al. model were conducted using the Gigaword dataset [53]. The same preprocessing steps of the data in [18] were performed in [52, 53]. Moreover, RCT also employed the Gigaword and DUC2004 datasets in experiments [54].

3.4. Evaluation and Results. Recall-Oriented Understudy for Gisting Evaluation 1 (ROUGE1), ROUGE2, and ROUGE-L were utilised to evaluate the Rush et al. model, and values of 28.18, 8.49, and 23.81, respectively, were obtained [18]. The experimental results of the Chopra et al. model showed that although DUC2004 was too complex for the experiments, on the Gigaword corpus, the proposed model outperformed state-of-the-art methods in terms of ROUGE1, ROUGE2, and ROUGE-L [39]. The values of ROUGE1, ROUGE2, and ROUGE-L were 28.97, 8.26, and 24.06, respectively. On the other hand, BLEU was employed to evaluate the Lopyrev model [29], while Khandelwal utilised perplexity [51]. The SEASS model was evaluated using ROUGE1, ROUGE2, and ROUGE-L, and the results of the three measures were 36.15, 17.54, and 33.63, respectively [52]. Moreover, ROUGE1, ROUGE2, and ROUGE-L were selected for evaluating the Cao et al. model [53]. The values of ROUGE1, ROUGE2, and ROUGE-L were 37.27, 17.65, and 34.24, respectively, and the results showed that fake summaries were reduced by 80%. In addition, the RCT was evaluated using ROUGE1, ROUGE2, and ROUGE-L with values 37.27, 18.19, and 34.62 compared with the Gigaword dataset. The results showed that the RCT model outperformed other models by generating a high-quality summary that contains silent information [54].

4. Multisentence Summary

In this section, multisentence summary and deep learning-based abstractive text summarisation are discussed. Multisentence summary methods include the get to the point method (summarisation with pointer-generator networks) [56], a deep reinforced model for abstractive summarization (RL) [57], generative adversarial network for abstractive text summarization [58], semantic phrase exploration (ATSDL) [30], bidirectional attention encoder-decoder and bidirectional beam search [35], key information guide network [59], text summarisation abstraction improvement [60], dual encoding for abstractive text summarisation (DEATS) [61], and abstractive document summarisation via bidirectional decoder (BiSum) [62], the text abstraction summary model based on BERT word embedding and RL [63], transformer-based model for single documents neural summarisation [64], text summarisation with pretrained encoders [65], and text summarisation method based on the double attention pointer network [49]. The pointer-generator [55] includes single-sentence and multisentence summaries. Additional details are presented in the following sections.

4.1. Abstractive Summarization Architecture

4.1.1. LSTM RN. A novel abstractive summarisation method was proposed in [56]; it generated a multisentence summary and addressed sentence repetition and inaccurate information. See et al. proposed a model that consists of a single-layer bidirectional LSTM encoder, a single-layer unidirectional LSTM decoder, and the sequence-to-sequence attention model proposed by [55]. The See et al. model generates a long text summary instead of headlines, which consists of one or two sentences. Moreover, the attention mechanism was employed, and the attention distribution facilitated the production of the next word in the summary by telling the decoder where to search in the source words, as shown in Figure 9. This mechanism constructed the weighted sum of the hidden state of the encoder that facilitated the generation of the context vector, where the context vector is the fixed size representation of the input. The probability (*Pvocab*) produced by the decoder was employed to generate the final prediction using the context vector and the decoder's last step. Furthermore, the value of *Pvocab* was equal to zero for OOV words. RL was employed for abstractive text summarisation in [57]. The proposed method in [57], which combined RL with supervised word prediction, was composed of a bidirectional LSTM-RNN encoder and a single LSTM decoder.

Two models—generative and discriminative models—were trained simultaneously to generate abstractive summary text using the adversarial process [58]. The maximum likelihood estimation (MLE) objective function employed in previous sequence-sequence models suffers from two problems: the difference between the training loss and the evaluation metric, and the unavailability of a golden token at testing time, which causes errors to accumulate during testing. To address the previous problems, the proposed approach exploited the adversarial framework. In the first step of the adversarial framework, reinforcement learning was employed to optimize the generator, which generates the summary from the original text. In the second step, the discriminator, which acts as a binary classifier, classified the summary as either a ground-truth summary or a machine-generated summary. The bidirectional LSTM encoder and attention mechanism were employed, as shown in [56].

Abstract text summarisation using the LSTM-CNN model based on exploring semantic phrases (ATSDL) was proposed in [30]. ATSDL is composed of two phases: the first phase extracts the phrases from the sentences, while the second phase learns the collocation of the extracted phrases using the LSTM model. To generate sentences that are general and natural, the input and output of the ATSDL model were phrases instead of words, and the phrases were divided into three main types, i.e., subject, relation, and object phrases, where the relation phrase represents the relation between the input phrase and the output phrase. The phrase was represented using a CNN layer. There are two main reasons for choosing the CNN: first, the CNN was efficient for sentence-level applications, and second, training was efficient since long-term dependency was unnecessary. Furthermore, to obtain several vectors for a phrase, multiple kernels with different widths that represent the dimensionality of the features were utilised. Within each kernel, the maximum feature was selected for each row in the kernel via maximum pooling. The resulting values were added to obtain the final value for each word in a phrase. Bidirectional LSTM was employed instead of a GRU on the encoder side since parameters are easy to tune with LSTM. Moreover, the decoder was divided into two modes: a generate mode and a copy mode. The generate mode generated the next phrase in the summary based on previously generated phrases and the hidden layers of the input on the encoder side, while the copy mode copied the phrase after the current input phrase if the current generated phrase was not suitable for the previously generated phrases in the summary. Figure 10 provides additional details.

Bidirectional encoder and decoder LSTM-RNNs were employed to generate abstractive multisentence summaries [35]. The proposed approach considered past and future context on the decoder side when making a prediction as it employed a bidirectional RNN. Using a bidirectional RNN on the decoder side addressed the problem of summary imbalance. An unbalanced summary could occur due to noise in a previous prediction, which will reduce the quality of all subsequent summaries. The bidirectional decoder consists of two LSTMs: the forward decoder and the backward decoder. The forward decoder decodes the information from left to right, while the backward decoder decodes the information from right to left. The last hidden state of the forward decoder is fed as the initial input to the backward decoder, and vice versa. Moreover, the researcher proposed a bidirectional beam-search method that generates summaries from the proposed bidirectional model. Bidirectional beam search combined information from the past and future to produce a better summary. Therefore, the

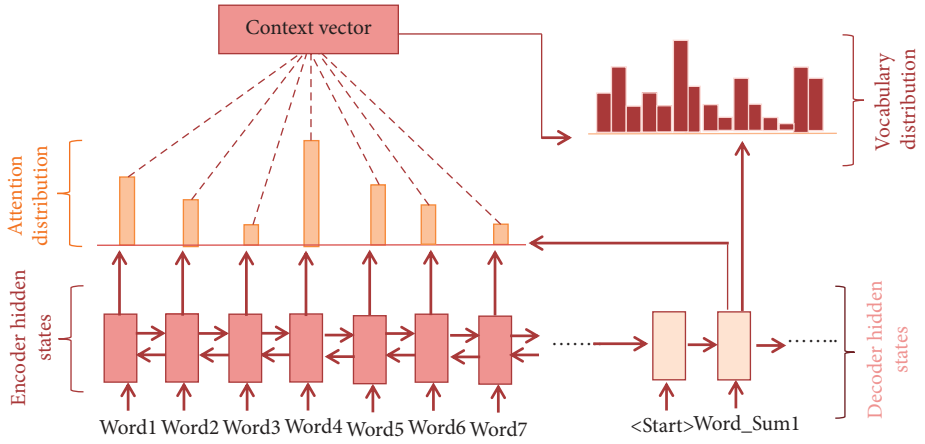

Figure 9: Baseline sequence-to-sequence model with attention mechanism [56].

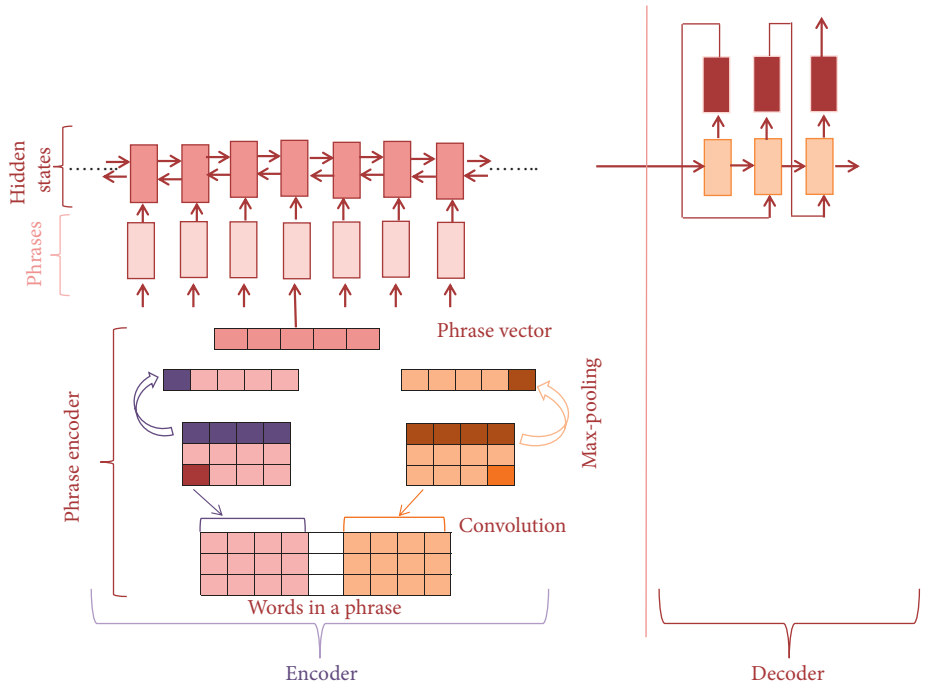

Figure 10: Semantic-unit-based LSTM model [30].

output summary was balanced by considering both past and future information and by using a bidirectional attention mechanism. In addition, the input sequence was read in reverse order based on the conclusion that LSTM learns better when reading the source in reverse order while remembering the order of the target [66, 67]. A softmax layer was employed on the decoder side to obtain the probability of each target word in the summary over the vocabulary distribution by taking the output of the decoder as input for the softmax layer. The decoder output depends on the internal representation of the encoder, i.e., the context vector, the current hidden state of the decoder, and the summary words previously generated by the decoder hidden states. The objective of training is to maximise the probability of the alignment between the sentence and the summary from both directions. During training, the input of the forward decoder is the previous reference summary token. However, during testing, the input of the forward decoder is the token generated in the previous step. The same situation is true for the backward decoder, where the input during training is the future token from the summary. Nevertheless, the bidirectional decoder has difficulty during testing since the complete summary must be known in advance; thus, the full backward decoder was generated and fed to the forward decoder using a unidirectional backward beam search.

A combination of abstractive and extractive methods was employed in the guiding generation model proposed by [59]. The extractive method generates keywords that are encoded by a key information guide network (KIGN) to represent key information. Furthermore, to predict the final summary of the long-term value, the proposed method applied a prediction guide mechanism [68]. A prediction

guide mechanism is a feedforward single-layer neural network that predicts the key information of the final summary during testing. The encoder-decoder architecture baseline of the proposed model is similar to that proposed by Nallapati et al. [55], where both the bidirectional LSTM encoder and the unidirectional LSTM decoder were employed. Both models applied the attention mechanism and softmax layer. Moreover, the process of generating the summary was improved by proposing KIGN, which considers as input the keywords extracted using the TextRank algorithm. In KIGN, key information is represented by concatenating the last forward hidden state and first backward hidden state. KIGN employs the attention mechanism and pointer mechanism. In general, the attention mechanism hardly identifies the keywords; thus, to identify keywords, the output of KIGN will be fed to the attention mechanism. As a result, the attention mechanism will be highly affected by the keywords. However, to enable the pointer network to identify the keywords, which are the output of KIGN, the encoder context vector and hidden state of the decoder will be fed to the pointer network, and the output will be employed to calculate the soft switch. The soft switch determines whether to copy the target from the original text or generate it from the vocabulary of the target, as shown in Figure 11.

The level of abstraction in the generated summary of the abstractive summarisation models was enhanced via the two techniques proposed in [60]: decoder decomposition and the use of a novel metric for optimising the overlap between the n-gram summary and the ground-truth summary. The decoder was decomposed into a contextual network and pretrained language model, as shown in Figure 12. The contextual network applies the source document to extract the relevant parts, and the pretrained language model is generated via prior knowledge. This decomposition method facilitates the addition of an external pretrained language model that is related to several domains. Furthermore, a novel metric was employed to generate an abstractive summary by including words that are not in the source document. Bidirectional LSTM was utilised in the encoder, and the decoder applied 3-layer unidirectional weight-dropped LSTM. In addition, the decoder utilised a temporal attention mechanism, which applied the intra-attention mechanism to consider previous hidden states. Furthermore, a pointer network was introduced to alternate between copying the output from the source document and selecting it from the vocabulary. As a result, the objective function combined between force learning and maximum likelihood.

A bidirectional decoder with a sequence-to-sequence architecture, which is referred to as BiSum, was employed to minimise error accumulation during testing [62]. Errors accumulate during testing as the input of the decoder is the previously generated summary word, and if one of the generated word summaries is incorrect, then the error will propagate through all subsequent summary words. In the bidirectional decoder, there are two decoders: a forward decoder and a backward decoder. The forward decoder generates the summary from left to right, while the backward decoder generates the summary from right to left. The forward decoder considers a reference from the backward decoder. However, there is only a single-layer encoder. The encoder and decoder employ an LSTM unit, but while the encoder utilises bidirectional LSTM, the decoders use unidirectional LSTM, as shown in Figure 13. To understand the summary generated by the backward decoder, the attention mechanism is applied in both the backward decoder and the encoder. Moreover, to address the problem of out-of-vocabulary words, an attention mechanism is employed in both decoders.

A double attention pointer network, which is referred to as (DAPT), was applied to generate an abstractive text summarisation model [49]. The encoder utilised bidirectional LSTM, while the decoder utilised unidirectional LSTM. The encoder key features were extracted using a self-attention mechanism. At the decoder, the beam search was employed. Moreover, more coherent and accurate summaries were generated. The repetition problem was addressed using an improved coverage mechanism with a truncation parameter. The model was optimised by generating a training model that is based on RL and scheduled sampling.

4.1.2. GRU-RNN. Dual encoding using a sequence-to-sequence RNN was proposed as the DEATS method [61]. The dual encoder consists of two levels of encoders, i.e., primary and secondary encoders, in addition to one decoder, and all of them employ a GRU. The primary encoder considers coarse encoding, while the secondary encoder considers fine encoding. The primary encoder and decoder are the same as the standard encoder-decoder model with an attention mechanism, and the secondary encoder generates a new context vector that is based on previous output and input. Moreover, an additional context vector provides meaningful information for the output. Thus, the repetition problem of the generated summary that was encountered in previous approaches is addressed. The semantic vector is generated on both levels of encoding: in the primary encoder, the semantic vector is generated for each input, while in the secondary encoder, the semantic vector is recalculated after the importance of each input word is calculated. The fixed-length output is partially generated at each stage in the decoder since it decodes in stages.

Figure 14 elaborates the DEATS process. The primary encoder produces a hidden state $h^p{}_j$ for each input j and content representation c^p. Next, the decoder decodes a fixed-length output, which is referred to as the decoder content representation c^d. The weight αj can be calculated using the hidden states $h^p{}_j$ and the content representations c^p and c^d. In this stage, the secondary encoder generates new hidden states or semantic context vectors $h^s{}_m$, which are fed to the decoder. Moreover, DEATS uses several advanced techniques, including a pointer-generator, copy mechanism, and coverage mechanism.

Wang et al. proposed a hybrid extractive-abstractive text summarisation model, which is based on combining the reinforcement learning with BERT word embedding [63]. In this hybrid model, a BERT feature-based strategy was used to

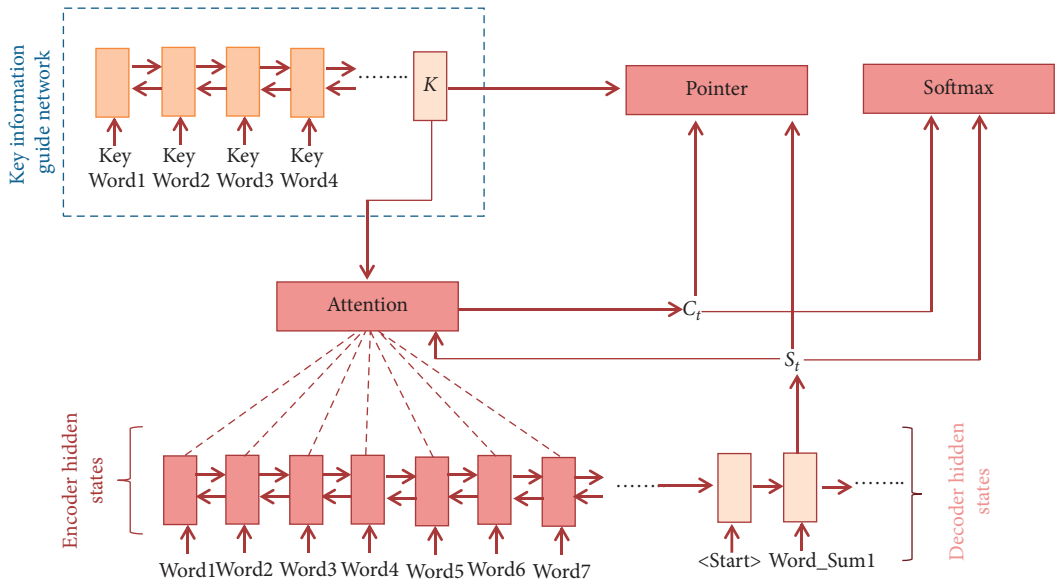

Figure 11: Key information guide network [59].

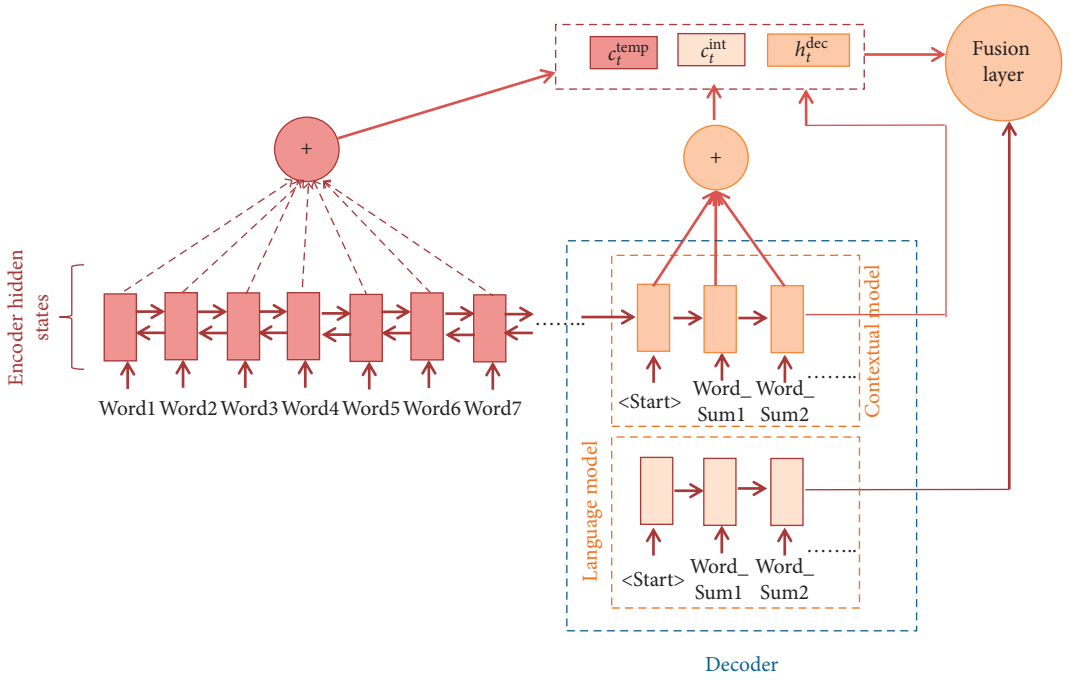

Figure 12: Decoder decomposed into a contextual model and a language model [60].

generate contextualised token embedding. This model consists of two submodels: abstractive agents and extractive agents, which are bridged using RL. Important sentences are extracted using the extraction model and rewritten using the abstraction model. A pointer-generator network was utilised to copy some parts of the original text, where the sentence-level and word-level attentions are combined. In addition, a beam search was performed at the decoder. In abstractive and extractive models, the encoder consists of a bidirectional GRU, while the decoder consists of a unidirectional GRU. The training process consists of pretraining and full training phases.

Egonmwan et al. proposed to use sequence-to-sequence and transformer models to generate abstractive summaries [64]. The proposed summarisation model consists of two modules: an extractive model and an abstractive model. The encoder transformer has the same architecture shown in [48]; however, instead of receiving the document representation as input, it receives sentence-level representation. The architecture of the abstractive model consists of a single-

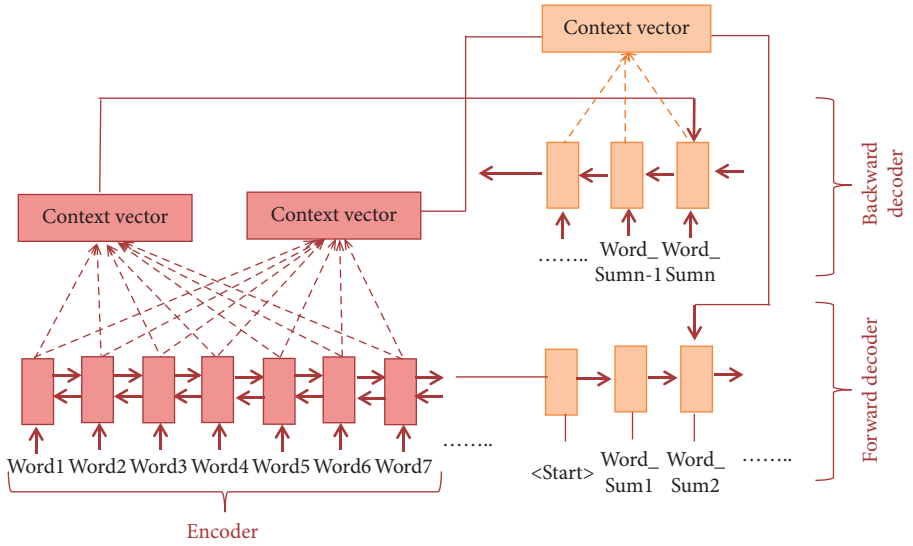

Figure 13: Abstractive document summarisation via bidirectional decoder (BiSum) [62].

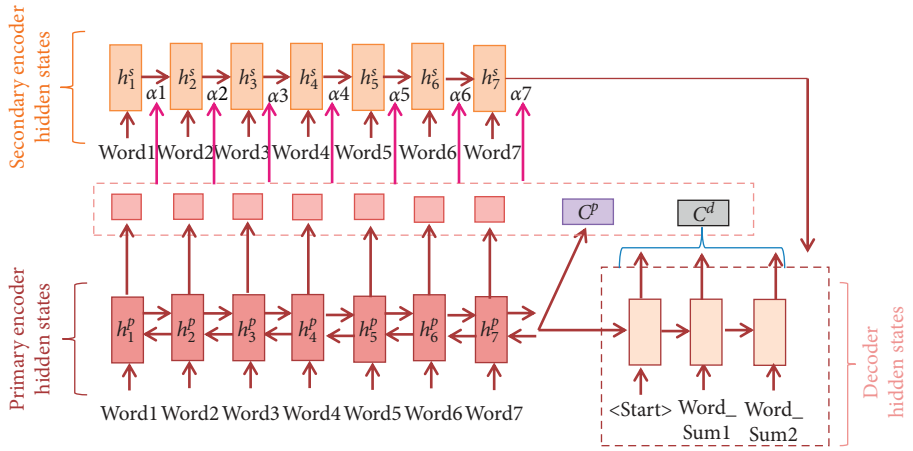

Figure 14: Dual encoding model [61].

layer unidirectional GRU at the encoder and single-layer unidirectional GRU at the decoder. The input of the encoder is the output of the transformer. A beam search was performed during inference at the decoder, while greedy-decoding was employed during training and validation.

4.1.3. Others. BERT is employed to represent the sentences of the document to express its semantic [65]. Liu et al. proposed abstractive and extractive summarisation models that are based on encoder-decoder architecture. The encoder used a BERT pretrained document-level encoder, while the decoder utilised a transformer that is randomly initialised and trained from scratch. In the abstractive model, the optimisers of the encoder and decoder are separated. Moreover, two stages of fine-tuning are utilised at the encoder: one stage in extractive summarisation and one stage in abstractive summarisation. At the decoder side, a beam search was performed; however, the coverage and copy mechanisms were not employed since these two mechanisms need additional tuning of the hyperparameters. The repetition problem was addressed by producing different summaries by using trigram-blocking. The OOV words rarely appear in the generated summary.

4.2. Word Embedding. The word embedding of the input for the See et al. model was learned from scratch instead of using a pretrained word embedding model [56]. On the other hand, both the input and output tokens applied the same embedding matrix *Wemb*, which was generated using the GloVe word embedding model in the Paulus et al. model [57]. Another word embedding matrix referred to as *Wout* was applied in the token generation layer. Additionally, a sharing weighting matrix was employed by both the shared embedding matrix *Wemb* and the *Wout* matrix. The sharing weighting matrixes improved the process of generating tokens since they considered the embedding syntax and semantic information.

The discriminator input sequence of the Liu et al. model was encoded using a maximum pooling CNN, where the result was passed to the softmax layer [58]. On the other hand, the word embedding that was applied in the Al-Sabahi et al. model was learned from scratch using the CNN/Daily Mail datasets with 128 dimensions [35]. Egonmwan et al. [64] used pretrained GloVe word embedding. BERT word embedding was utilised in the models proposed by Wang et al. [63] and Liu et al. [65].

4.3. Dataset and Dataset Preprocessing. Experiments were conducted with the See et al. [56], Al-Sabahi et al. [35], and Li et al. [59] models using CNN/Daily Mail datasets, which consist of 781 tokens paired with 56 tokens on average; 287,226 pairs, 13,368 pairs, and 11,490 pairs were utilised for training, validation, and testing, respectively [56]. In the model proposed by Paulus et al., the document was preprocessed using the same method applied in [55]. The proposed model was evaluated using two datasets: the CNN/Daily News dataset and the New York Times dataset. The CNN/Daily Mail dataset was utilised by Liu et al. for training their model [58].

The ATSDL model consisted of three stages: text preprocessing, phrase extractions, and summary generation [30]. During text preprocessing, the CoreNLP tool was employed to segment the words, reduce the morphology, and resolve the coreference. The second stage of the ATSDL model was phrase extraction, which included the acquisition, refinement and combination of phrases. In addition, multiorder semantic parsing (MOSP), which was proposed to create multilayer binary semantics, was applied for phrase extraction. The first step of MOSP was to perform Stanford NLP parsing, which is a specialised tool that retrieved the lexical and syntactic features from the preprocessed sentences. Next, dependency parsing was performed to create a binary tree by determining the root of the tree, which represents the relational phrase. If the child node has children, then the child is considered a new root with children; this process continues recursively until there are no children for the root. In this case, the tree structure is completed. Accordingly, the compound phrases can be explored via dependency parsing. However, one of the important stages of phrase extraction is refinement, during which redundant and incorrect phrases are refined before training by applying simple rules. First, the phrase triples at the topmost level are exploited since they carry the most semantic information. Second, triple phrases with subject and object phrases and no nouns are deleted since the noun contains a considerable amount of conceptual information. Triple phrases without a verb in a relational phrase are deleted. Moreover, phrase extraction includes phrase combination, during which phrases with the same meaning are combined to minimise redundancy and the time required to train the LSTM-RNN. To achieve the goal of the previous task and determine whether two phrases can be combined, a set of artificial rules are applied. The experiments were conducted using the CNN and Daily Mail datasets, which consisted of 92,000 text sources and 219,000 text sources, respectively.

The Kryściński et al. [60] model was trained using a CNN/Daily Mail dataset, which was preprocessed using the method from 56[55, 56]. The experiments of DEATS were conducted using the CNN/Daily Mail dataset and DUC2004 corpus [61]. The experiments of the BiSum model were performed using the CNN/Daily Mail dataset [62]. In the Wang et al. proposed model, CNN/Daily Mail and DUC2002 were employed in experiments [63] while the Egonmwan et al. model employed the CNN/Daily and Newsroom datasets in experiments [64]. Experiments were conducted with the Liu et al. [65] model using three benchmark datasets, including CNN/Daily Mail, New York Times Annotated Corpus (NYT), and XSum. Experiments were also conducted with the DAPT model using the CNN/Daily Mail and LCSTS datasets [49].

4.4. Evaluation and Results. The evaluation metrics ROUGE1, ROUGE2, and ROUGE-L, with values of 39.53, 17.28, and 36.38, respectively, were applied to measure the performance of the See et al. model [56], which outperformed previous approaches by at least two points in terms of the ROUGE metrics. Reinforcement learning with the intra-attention model achieved the following results: ROUGE1, 41.16; ROUGE2, 15.75; and ROUGE-L, 39.08 [57]. The results for the maximum likelihood model were 39.87, 15.82, and 36.9 for ROUGE1, ROUGE2, and ROUGE-L, respectively. Overall, the proposed approach yielded high-quality generated summaries [57].

ROUGE1, ROUGE2, and ROUGE-L were utilised to evaluate the Liu et al. model, which obtained values of 39.92, 17.65, and 36.71, respectively [58]. In addition, a manual qualitative evaluation was performed to evaluate the quality and readability of the summary. Two participants evaluated the summaries of 50 test examples that were selected randomly from the datasets. Each summary was given a score from 1 to 5, where 1 indicates a low level of readability and 5 indicates a high level of readability.

ROUGE1 and ROUGE2 were used to evaluate the ATSDL model [30]. The value of ROUGE1 was 34.9, and the value of ROUGE2 was 17.8. Furthermore, ROUGE1, ROUGE2, and ROUGE-L were applied as evaluation metrics of the Al-Sabahi et al. and Li et al. models, and the values of 42.6, 18.8, and 38.5, respectively, were obtained for the Al-Sabahi et al. model [35], while the values of 38.95, 17.12, and 35.68, respectively, were obtained for the Li et al. model [58].

The evaluation of the Kryściński et al. model was conducted using quantitative and qualitative evaluations [60]. The quantitative evaluations included ROUGE1, ROUGE2, and ROUGE-L, and the values of 40.19, 17.38, and 37.52, respectively, were obtained. Additionally, a novel score related to the n-gram was employed to measure the level of abstraction in the summary. The qualitative evaluation involved the manual evaluation of the proposed model. Five participants evaluated 100 full-text summaries in terms of relevance and readability by giving each document a value from 1 to 10. Furthermore, for comparison purposes, full-text summaries from two previous studies [56, 58] were selected. The evaluators graded the output summaries without knowing which model generated them.

Moreover, ROUGE1, ROUGE2, and ROUGE-L were applied for evaluating DEATS, and the values of 40.85, 18.08, and 37.13, respectively, were obtained for the CNN/Daily Mail dataset [61]. The experimental results of the BiSum model showed that the values of ROUGE1, ROUGE2, and ROUGE-L were 37.01, 15.95, and 33.66, respectively [62].

Several variations in the Wang et al. model were implemented. The best results were achieved by the BEAR (large + WordPiece) model, where the WordPiece tokeniser was utilised. The values of ROUGE1, ROUGE2, and ROUGE-L were 41.95, 20.26, and 39.49, respectively [63]. In Egonmwan et al. model, the values of ROUGE1 and ROUGE2 were 41.89 and 18.90, respectively, while the value of ROUGE3 was 38.92. Several variations in the Liu et al. [65] model were evaluated using ROUGE1, ROUGE2, and ROUGE-L, where the best model, which is referred to as BERTSUMEXT (large), achieved the values of 43.85, 20.34, and 39.90 for ROUGE1, ROUGE2, and ROUGE-L, respectively, over the CNN/Daily Mail datasets. Moreover, the model was evaluated by a human via a question and answering paradigm, where 20 documents were selected for evaluation. Three values were chosen for evaluating the answer: a score of 1 indicates the correct answer; a score of 0.5 indicates a partially correct answer; and a score of 0 indicates a wrong answer. ROUGE1, ROUGE2, and ROUGE-L for the DAPT model over the CNN/Daily Mail datasets were 40.72, 18.28, and 37.35, respectively.

Finally, the pointer-generator approach was applied on the single-sentence and multisentence summaries. Attention encoder-decoder RNNs were employed to model the abstractive text summaries [55]. Both the encoder and decoder have the same number of hidden states. Additionally, the proposed model consists of a softmax layer for generating the words based on the vocabulary of the target. The encoder and decoder differ in terms of their components. The encoder consists of two bidirectional GRU-RNNs—a GRU-RNN for the word level and a GRU-RNN for the sentence level—while the decoder uses a unidirectional GRU-RNN, as shown in Figure 15. Furthermore, the decoder uses batching, where the vocabulary at the decoder for each minibatch is restricted to the words in the batch of the source document. Instead of considering every vocabulary, only certain vocabularies were added based on the frequency of the vocabulary in the target dictionary to decrease the size of the decoder softmax layer. Several linguistic features were considered in addition to the word embedding of the input words to identify the key entities of the document. Linguistic and statistical features included TF-IDF statistics and the part-of-speech and named-entity tags of the words. Specifically, the part-of-speech tags were stored in matrixes for each tag type that was similar to word embedding, while the TF-IDF feature was discretised in bins with a fixed number, where one-hot representation was employed to represent the value of the bins. The one-hot matrix consisted of the number of bin entries, where only one entry was set to one to indicate the value of the TF-IDF of a certain word. This process permitted the TF-IDF to be addressed in the same way as any other tag by concatenating all the embeddings into one long vector, as shown in Figure 16. The experiments were conducted using the annotated Gigaword corpus with 3.8 M training examples, the DUC corpus, and the CNN/Daily Mail corpus. The preprocessing methods included tokenisation and part-of-speech and name-entity generation. Additionally, the Word2Vec model with 200 dimensions was applied for word embedding and trained using the Gigaword corpus. Additionally, the hidden states had 400 dimensions in both the encoder and the decoder. Furthermore, datasets with multisentence summaries were utilised in the experiments. The values of ROUGE1, ROUGE2, and ROUGE-L were higher than those of previous work on abstractive summarisation, with values of 35.46, 13.3, and 32.65, respectively.

Finally, for both single-sentence summary and multisentence summary models, the components of the encoder and decoder of each approach are displayed in Table 1. Furthermore, dataset preprocessing and word embedding of several approaches are appeared in Table 2 while training, optimization, mechanism, and search at the decoder are presented in Table 3.

5. Datasets for Text Summarization

Various datasets were selected for abstractive text summarisation, including DUC2003, DUC2004 [69], Gigaword [70], and CNN/Daily Mail [71]. The DUC datasets are produced for the Document Understanding Conference; although their quality is high, they are small datasets that are typically employed to evaluate summarisation models. The DUC2003 and DUC2004 datasets consist of 500 articles. The Gigaword dataset from the Stanford University Linguistics Department was the most common dataset for model training in 2015 and 2016. Gigaword consists of approximately 10 million documents from seven news sources, including the New York Times, Associated Press, and Washington Post. Gigaword is one of the largest and most diverse summarisation datasets even though it contains headlines instead of summaries; thus, it is considered to contain single-sentence summaries.

Recent studies utilised the CNN/Daily Mail datasets for training and evaluation. The CNN/Daily Mail datasets consist of bullet points that describe the articles, where multisentence summaries are created by concatenating the bullet points of the article [5]. CNN/Daily Mail datasets that are applied in abstractive summarisation were presented by Nallapati et al. [55]. These datasets were created by modifying the CNN/Daily Mail datasets that were generated by Hermann et al. [71]. The Hermann et al. datasets were utilised for extractive summarisation. The abstractive summarisation CNN/Daily Mail datasets have 286,817 pairs for training and 13,368 pairs for validation, while 11,487 pairs were applied in testing. In training, the source documents have 766 words (on average 29.74 sentences), while the summaries have 53 words (on average 3.72 sentences) [55].

In April 2018, NEWSROOM, a summarisation dataset that consists of 1.3 million articles collected from social media metadata from 1998 to 2017, was produced [72]. The

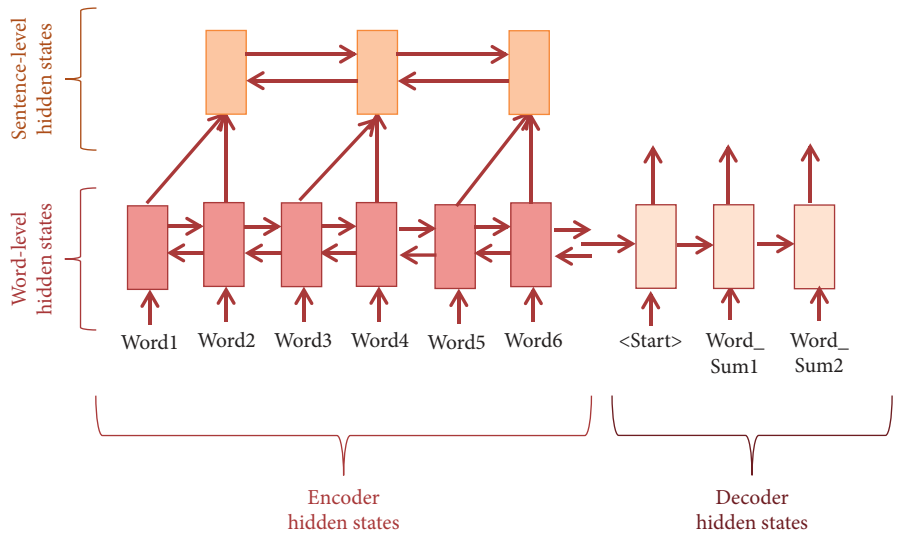

Figure 15: Word-level and sentence-level bidirectional GRU-RNN [55].

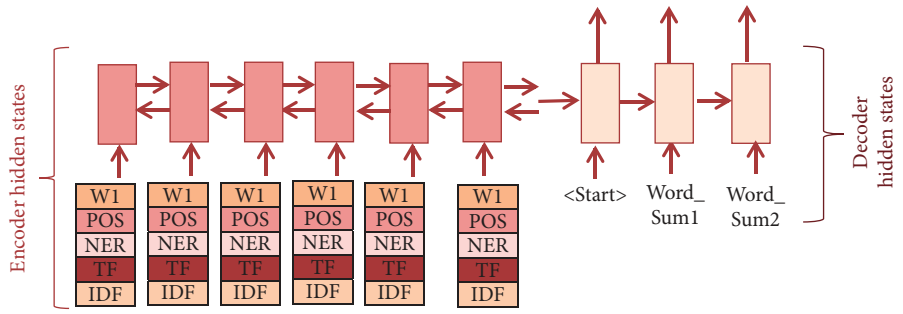

Figure 16: Word embedding concatenated with discretized TF-IDF, POS, and NER one-embedding vectors [55].

Table 1: Encoder and decoder components.

Reference	Year	Encoder	Decoder
[18]	2015	Bag-of-words, convolutional, and attention-based	
[29]	2015	RNN with LSTM units and attention	RNN with LSTM units and attention
[39]	2016	RNN-LSTM decoder RNN	Word-based
[50]	2016	GRU + QRNN + attention	GRU + RNN QRNN
[38]	2016	Unidirectional RNN attentive encoder-decoder LSTM	Unidirectional RNN attentive encoder-decoder LSTM
		Bidirectional LSTM	Unidirectional LSTM
		Bidirectional LSTM	Decoder that had global attention
[51]	2016	LSTM-RNN	LSTM-RNN
[55]	2016	Two bidirectional GRU-RNN	GRU-RNN unidirection
[52]	2017	Bidirectional GRU	Unidirectional GRU
[53]	2017	Bidirectional GRU	Unidirectional GRU
[56]	2017	Single-layer bidirectional LSTM + attention	Single-layer unidirectional LSTM
[57]	2017	Bidirectional LSTM-RNN + intra-attention single	LSTM decoder + intra-attention
[58]	2018	Bidirectional LSTM	Unidirectional LSTM
[30]	2018	Bidirectional LSTM	Unidirectional LSTM
[35]	2018	Bidirectional LSTM	Bidirectional LSTM
[59]	2018	Bidirectional LSTM	Unidirectional LSTM
[60]	2018	Bidirectional LSTM	3-layer unidirectional LSTM
[61]	2018	Bidirectional GRU	Unidirectional GRU
[62]	2018	Bidirectional LSTM	Two-decoder unidirectional LSTM
[63]	2019	Bidirectional GRU	Unidirectional GRU
[64]	2019	Unidirectional GRU	Unidirectional GRU
[49]	2020	Bidirectional LSTM	Unidirectional LSTM

Table 2: Dataset preprocessing and word embedding.

Reference	Authors	Dataset preprocessing	Input (word embedding)
[18]	Rush et al.	PTB tokenization by using "#" to replace all digits, converting all letters to lower case, and "UNK" to replace words that occurred fewer than 5 times	Bag-of-words of the input sentence embedding
[39]	Chopra et al.	PTB tokenization by using "#" to replace all digits, converting all letters to lower case, and "UNK" to replace words that occurred fewer than 5 times	Encodes the position information of the input words
[55]	Nallapati et al.	Part-of-speech and name-entity tags generating and tokenization	(i) Encodes the position information of the input words (ii) The input text was represented using the Word2Vec model with 200 dimensions that was trained using Gigaword corpus (iii) Continuous features such as TF-IDF were represented using bins and one-hot representation for bins (iv) Lookup embedding for part-of-speech tagging and name-entity tagging
[52]	Zhou et al.	PTB tokenization by using "#" to replace all digits, converting all letters to lower case, and "UNK" to replace words that occurred fewer than 5 times	Word embedding with size equal to 300
[53]	Cao et al.	Normalization and tokenization, using the "#" to replace digits, convert the words to lower case, and "UNK" to replace the least frequent words.	GloVe word embedding with dimension size equal to 200
[54]	Cai et al.	Byte pair encoding (BPE) was used in segmentation	Transformer
[50]	Adelson et al.	Converting the article and their headlines to lower case letters	GloVe word embedding
[29]	Lopyrev	Tokenization, converting the article and their headlines to lower case letters, using the symbol ⟨unk⟩ to replace rare words	The input was represented using the distributed representation
[38]	Jobson et al.		The word embedding randomly initialised and updated during training while GloVe word embedding was used to represent the words in the second and third models
[56]	See et al.		The word embedding of the input for was learned from scratch instead of using a pretrained word embedding model
[57]	Paulus et al.	The same as in [55]	GloVe
[58]	Liu et al.		CNN maximum pooling was used to encode the discriminator input sequence
[30]	Song et al.	The words were segmented using CoreNLP tool, resolving the coreference and reducing the morphology	Convolutional neural network was used to represent the phrases
[35]	Al-Sabahi et al.		The word embedding is learned from scratch during training with a dimension of 128
[59]	Li et al.	The same as in [55]	Learned from scratch during training
[60]	Kryściński et al.	The same as in [55]	Embedding layer with a dimension of 400
[61]	Yao et al.		The word embedding is learned from scratch during training with a dimension of 128
[62]	Wan et al.	No word segmentation	Embedding layer learned during training
[65]	Liu et al.		BERT
[63]	Wang et al.	Using WordPiece tokenizer	BERT
[64]	Egonmwan et al.		GloVe word embedding with dimension size equal to 300

NEWSROOM dataset consists of 992,985 pairs for training and 108,612 and 108,655 pairs for validation and testing, respectively [22]. The quality of the summaries is high, and the style of the summarisation is diverse. Figure 17 displays the number of surveyed papers that applied each of the datasets. Nine research papers utilised Gigaword, fourteen papers employed the CNN/Daily Mail datasets (largest number of papers on the list), and one study applied the ACL Anthology Reference, DUC2002, DUC2004, New York Times Annotated Corpus (NYT), and XSum datasets.

Table 3: Training, optimization, mechanism, and search at the decoder.

Reference	Authors	Training and optimization	Mechanism	Search at decoder (siz)
[18]	Rush et al.	Stochastic gradient descent to minimise negative log-likelihood		Beam search
[39]	Chopra et al.	Minimizing negative log-likelihood using end-to-end using stochastic gradient descent	Encodes the position information of the input words	Beam search
[55]	Nallapati et al.	Optimize the conditional likelihood using Adadelta	Pointer mechanism	Beam search (5)
[52]	Zhou et al.	Stochastic gradient descent, Adam optimizer, optimizing the negative log-likelihood	Attention mechanism	Beam search (12)
[53]	Cao et al.	Adam optimizer, optimizing the negative log-likelihood	Copy mechanism, coverage mechanism, dual-attention decoder	Beam search (6)
[54]	Cai et al.	Cross entropy is used as the loss function	Attention mechanism	Beam search (5)
[50]	Adelson et al.	Adam	Attention mechanism	
[29]	Lopyrev	RMSProp adaptive gradient method	Simple and complex attention mechanism	Beam search
[38]	Jobson et al.	Adadelta, minimising the negative log probability of prediction word	Bilinear attention mechanism, pointer mechanism	
[56]	See et al.	Adadelta	Coverage mechanism, attention mechanism, pointer mechanism	Beam search (4)
[57]	Paulus et al.	Adam, RL	Intradecoder attention mechanism, pointer mechanism, copy mechanism, RL	Beam search (5)
[58]	Liu et al.	Adadelta stochastic gradient descent	Attention mechanism, pointer mechanism, copy mechanism, RL	
[30]	Song et al.		Attention mechanism, copy mechanism	
[35]	Al-Sabahi et al.	Adagrad	Pointer mechanism, coverage mechanism, copy mechanism	Bidirectional beam search
[59]	Li et al.	Adadelta	Attention mechanism, pointer mechanism, copy mechanism, prediction guide mechanism	Beam search
[60]	Kryściński et al.	Asynchronous gradient descent optimizer	Temporal attention and intra-attention pointer mechanism, RL	Beam search
[61]	Yao et al.	RL, Adagrad	Attention mechanism, pointer mechanism, copy mechanism, coverage mechanism, RL	Beam search (4)
[62]	Wan et al.	Adagrad	Attention mechanism, pointer mechanism	Beam-search backward (2) and forward (4)
[65]	Liu et al.	Adam	Self-attention mechanism	Beam search (5)
[63]	Wang et al.	Gradient of reinforcement learning, Adam, cross-entropy loss function	Attention mechanism, pointer mechanism, copy mechanism, new coverage mechanism	Beam search
[64]	Egonmwan et al.	Adam	Self-attention mechanism	Greedy-decoding during training and validation. Beam search at decoding during testing
[49]	Peng et al.	Adam, gradient descent, cross-entropy loss	Coverage mechanism, RL, double attention pointer network (DAPT)	Beam search (5)

Table 4 lists the datasets that are used to train and validate the summarisation methods in the research papers listed in this work.

6. Evaluation Measures

The package ROUGE is employed to evaluate the text summarisation techniques by comparing the generated summary with a manually generated summary [73]. The package consists of several measures to evaluate the performance of text summarisation techniques, such as ROUGE-N (ROUGE1 and ROUGE2) and ROUGE-L, which were employed in several studies [38]. ROUGE-N is n-gram recall such that ROUGE1 and ROUGE2 are related to unigrams and bigrams, respectively, while ROUGE-L is related to the longest common substring. Since the manual

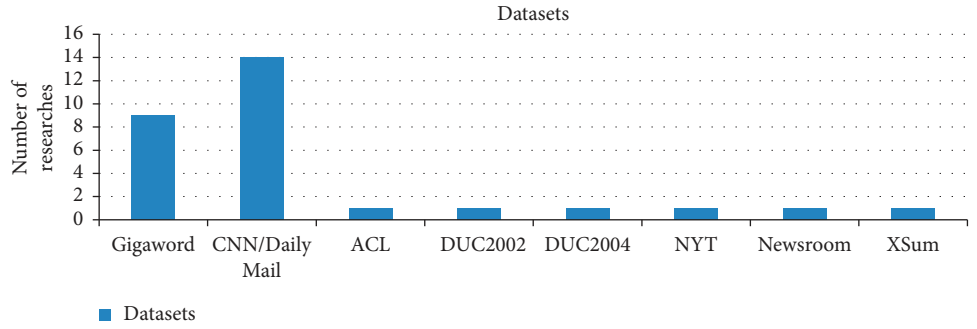

Figure 17: The number of research papers that used the Gigaword, CNN/Daily Mail, ACL DUC2002, DUC2004, NYT, Newsroom, and XSum datasets [61].

Table 4: Abstractive summarisation datasets.

Reference	Training	Summarization Evaluation
[18]	Gigaword	DUC2003 and DUC2004
[39]	Gigaword	DUC2004
[50]	Gigaword	Gigaword
[29]	Gigaword	Articles from BBC, The Wall Street Journal, Guardian, Huffington Post, and Forbes
[38]	Gigaword	—
[54]	Gigaword and DUC2004	Gigaword and DUC2004
[51]	ACL anthology reference	ACL anthology reference
[52]	Gigaword and DUC2004	Gigaword and DUC2004
[53]	Gigaword and DUC2004	Gigaword and DUC2004
[56]	CNN/Daily Mail	CNN/Daily Mail
[57]	CNN/Daily and New York Times	CNN/Daily and New York Times
[58]	CNN/Daily Mail	CNN/Daily Mail
[30]	CNN/Daily Mail	CNN/Daily Mail
[35]	CNN/Daily Mail	CNN/Daily Mail
[59]	CNN/Daily Mail	CNN/Daily Mail
[60]	CNN/Daily Mail	CNN/Daily Mail
[61]	CNN/Daily Mail	CNN/Daily Mail
[55]	Gigaword DUC CNN/Daily Mail	Gigaword DUC CNN/Daily Mail
[62]	CNN/Daily Mail	CNN/Daily Mail
[65]	CNN/Daily Mail, NYT, and XSum	CNN/Daily Mail, NYT, and XSum
[63]	CNN/Daily Mail and DUC2002	CNN/Daily Mail and DUC2002
[64]	CNN/Daily Mail and Newsroom	CNN/Daily Mail and Newsroom
[49]	CNN/Daily Mail	CNN/Daily Mail

evaluation of automatic text summarisation is a time-consuming process and requires extensive effort, ROUGE is employed as a standard for evaluating text summarisation. ROUGE-N is calculated using the following equation:

$$\text{ROUGE} - N = \frac{\sum S \in \{\text{REFERENCE SUMMARIES}\} \sum \text{gram}_n \in \text{Count}_{\text{match}}(\text{gram}_n)}{\sum S \in \{\text{REFERENCE SUMMARIES}\} \sum \text{gram}_n \in \text{Count}(\text{gram}_n)}, \tag{1}$$

where S is the reference summary, n is the n-gram length, and $\text{Count}_{\text{match}}(\text{gram}_n)$ is the maximum number of matching n-gram words between the reference summary and the generated summary. Count (gram_n) is the total number of n-gram words in the reference summary [73].

ROUGE-L is the longest common subsequence (LCS), which represents the maximum length of the common matching words between the reference summary and the generated summary. LCS calculation does not necessarily require the match words to be consecutive; however, the order of occurrence is important. In addition, no predefined number of match words is required. LCS considers only the main in-sequence, which is one of its disadvantages since the final score will not include other matches. For example, assume that the reference summary R and the automatic summary A are as follows:

R: Ahmed ate the apple.

A: the apple Ahmed ate.

In this case, ROUGE-L will consider either "Ahmed ate" or "the apple" but not both, similar to LCS.

Tables 5 and 6 present the values of ROUGE1, ROUGE2, and ROUGE-L for the text summarisation methods in the various studies reviewed in this research. In addition, Perplexity was employed in [18, 39, 51], and BLEU was utilised in [29]. The models were evaluated using various datasets. The other models applied ROUGE1, ROUGE2, and ROUGE-L for evaluation. It can be seen that the highest values of ROUGE1, ROUGE2, and ROUGE-L for text summarisation with the pretrained encoder model were 43.85, 20.34, and 39.9, respectively [65]. Even though ROUGE was employed to evaluate abstractive summarisation, it is better to obtain new methods to evaluate the quality of summarisation. The new evaluation metrics must consider novel words and semantics since the generated summary contains words that do not exist in the original text. However, ROUGE was very suitable for extractive text summarisation.

Based on our taxonomy, we divided the results of ROUGE1, ROUGE2, and ROUGE-L into two groups. The first group considered single-sentence summary approaches, while the second group considered multisentence summary approaches. Figure 18 compares several deep learning techniques in terms of ROUGE1, ROUGE2, and ROUGE-L for the Gigaword datasets, consisting of single-sentence summary documents. The highest values for ROUGE1, ROUGE2, and ROUGE-L were achieved by the RCT model [54]. The values for ROUGE1, ROUGE2, and ROUGE-L were 37.27, 18.19, and 34.62, respectively.

Furthermore, Figure 19 compares the ROUGE1, ROUGE2, and ROUGE-L values for abstractive text summarisation methods for the CNN/Daily Mail datasets, which consist of multisentence summary documents. The highest values of ROUGE1, ROUGE2, and ROUGE-L were achieved for text summarisation with a pretrained encoder model. The values for ROUGE1, ROUGE2, and ROUGE-L were 43.85, 20.34, and 39.9, respectively [65]. It can be clearly seen that the best model in the single-sentence summary and multisentence summary is the models that employed BERT word embedding and were based on transformers. The ROUGE values for the CNN/Daily Mail datasets are larger than those for the Gigaword dataset, as Gigaword is utilised for single-sentence summaries as it contains headlines that are treated as summaries, while the CNN/Daily Mail datasets are multisentence summaries. Thus, the summaries in the CNN/Daily Mail datasets are longer than the summaries in Gigaword.

Liu et al. selected two human elevators to evaluate the readability of the generated summary of 50 test examples of 5 models [58]. The value of 5 indicates that the generated summary is highly readable, while the value of 1 indicates that the generated summary has a low level of readability. It can be clearly seen from the results that the Liu et al. model was better than the other four models in terms of ROUGE1, ROUGE2, and human evaluation, even though the model is not optimal with respect to the ROUGE-L value. In addition to quantitative measures, qualitative evaluation measures are important. Kryściński et al. also performed qualitative evaluation to evaluate the quality of the generated summary [60]. Five human evaluators evaluated the relevance and readability of 100 randomly selected test examples, where two values are utilised: 1 and 10. The value of 1 indicates that the generated summary is less readable and less relevance while the value of 10 indicates that the generated summary is readable and very relevance. The results showed that, in terms of readability, the model proposed by Kryściński et al. is slightly inferior to See et al. [56] and Liu et al. [58] models with mean values of 6.35, 6.76, and 6.79 for Kryściński et al., to See et al. and Liu et al., respectively. On the other hand, with respect to the relevance, the means values of the three models are relevance with values of 6.63, 6.73, and 6.74 for Kryściński et al., to See et al. and Liu et al., respectively. However, the Kryściński et al. model was the best in terms of ROUGE1, ROUGE2, and ROUGE-L.

Liu et al. evaluated the quality of the generated summary in terms of succinctness, informativeness, and fluency in addition to measuring the level of retaining key information, which was achieved by human evaluation [65]. In addition, qualitative evaluation evaluated the output in terms of grammatical mistakes. Three values were selected for evaluating 20 test examples: 1 indicates a correct answer, 0.5 indicates a partially correct answer, and 0 indicates an incorrect answer. We can conclude that quantitative evaluations, which include ROUGE1, ROUGE2, and ROUGE-L, are not enough for evaluating the generated summary of abstractive text summarisation, especially when measuring readability, relevance, and fluency. Therefore, qualitative measures, which can be achieved by manual evaluation, are very important. However, qualitative measures without quantitative measures are not enough due to the small number of testing examples and evaluators.

7. Challenges and Solutions

Text summarisation approaches have faced various challenges; although some have been solved, others still need to be addressed. In this section, these challenges and their possible solutions are discussed.

7.1. Unavailability of the Golden Token during Testing. Due to the availability of golden tokens (i.e., reference summary tokens) during training, previous tokens in the headline can be input into the decoder at the next step. However, during testing, the golden tokens are not available; thus, the input for the next step in the decoder will be limited to the previously generated output word. To solve this issue, which becomes more challenging when addressing small datasets, different solutions have been proposed. For example, in reference [51], the data-as-demonstrator (DaD) model [74] is utilised. In DaD, at each step, based on a coin flip, either a gold token is utilised during training or the previous step is employed during both testing and training. In this manner, at least the training step receives the same

Table 5: Evaluation measures of several deep learning abstractive text summarisation methods over the Gigaword dataset.

Reference	Year	Authors	Model	ROUGE1	ROUGE2	ROUGE-L
[18]	2015	Rush et al.	ABS+	28.18	8.49	23.81
[39]	2016	Chopra et al.	RAS-Elman ($k = 10$)	28.97	8.26	24.06
[55]	2016	Nallapati et al.	Words-lvt5k-1sent	28.61	9.42	25.24
[52]	2017	Zhou et al.	SEASS	36.15	17.54	33.63
[53]	2018	Cao et al.	FTSumg	37.27	17.65	34.24
[54]	2019	Cai et al.	RCT	37.27	18.19	34.62

Table 6: Evaluation measures of several abstractive text summarisation methods over the CNN/Daily Mail datasets.

Reference	Year	Authors	Model	ROUGE1	ROUGE2	ROUGE-L
[55]	2016	Nallapati et al.	Words-lvt2k-temp-att	35.46	13.30	32.65
[56]	2017	See et al.	Pointer-generator + coverage	39.53	17.28	36.38
[57]	2017	Paulus et al.	Reinforcement learning, with intra-attention	41.16	15.75	39.08
[57]	2017	Paulus et al.	Maximum-likelihood + RL, with intra-attention	39.87	15.82	36.90
[58]	2018	Liu et al.	Adversarial network	39.92	17.65	36.71
[30]	2018	Song et al.	ATSDL	34.9	17.8	—
[35]	2018	Al-Sabahi et al.	Bidirectional attentional encoder-decoder	42.6	18.8	38.5
[59]	2018	Li et al.	Key information guide network	38.95	17.12	35.68
[60]	2018	Kryściński et al.	ML + RL ROUGE + Novel, with LM	40.19	17.38	37.52
[61]	2018	Yao et al.	DEATS	40.85	18.08	37.13
[62]	2018	Wan et al.	BiSum	37.01	15.95	33.66
[63]	2019	Wang et al.	BEAR (large + WordPiece)	41.95	20.26	39.49
[64]	2019	Egonmwan et al.	TRANS-ext + filter + abs	41.89	18.9	38.92
[65]	2020	Liu et al.	BERTSUMEXT (large)	43.85	20.34	39.90
[49]	2020	Peng et al.	DAPT + imp-coverage (RL + MLE (ss))	40.72	18.28	37.35

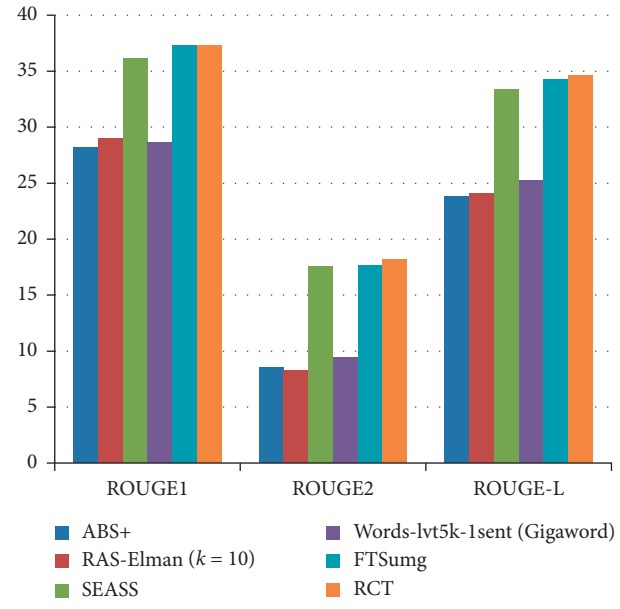

Figure 18: ROUGE1, ROUGE2, and ROUGE-L scores of several deep learning abstractive text summarisation methods for the Gigaword dataset.

input as testing. In all cases, the first input of the decoder is the ⟨EOS⟩ token, and the same calculations are applied to compute the loss. In [29], teacher forcing is employed to address this challenge: during training, instead of feeding the expected word from the headline, 10% of the time, the generated word of the previous step is fed back [75, 76]. Moreover, the mass convolution of the QRNN is applied in [50] since the dependency of words generated in the future is difficult to determine.

7.2. Out-of-Vocabulary (OOV) Words. One of the challenges that may occur during testing is that the central words of the test document may be rare or unseen during training; these words are referred to as OOV words. In 61[55, 61], a switching decoder/pointer was employed to address OOV words by using pointers to point to their original positions in the source document. The switch on the decoder side is used to alternate between generating a word and using a pointer, as shown in Figure 20 [55]. When the switch is turned off, the decoder will use the pointer to point to the word in the source to copy it to the memory. When the switch is turned on, the decoder will generate a word from the target vocabularies. Conversely, researchers in [56] addressed OOV words via probability generation *Pgen*, where the value is calculated from the context vector and decoder state, as shown in Figure 21. To generate the output word, *Pgen* switches between copying the output words from the input sequence and generating them from the vocabulary. Furthermore, the pointer-generator technique is applied to point to input words to copy them. The combination between the words in the input and the vocabulary is referred to the extended vocabulary. In addition, in [57], to generate the tokens on the decoder side, the decoder utilised the switch function at each timestep to switch between generating the token using the softmax layer and using the pointer mechanism to point to the input sequence position for

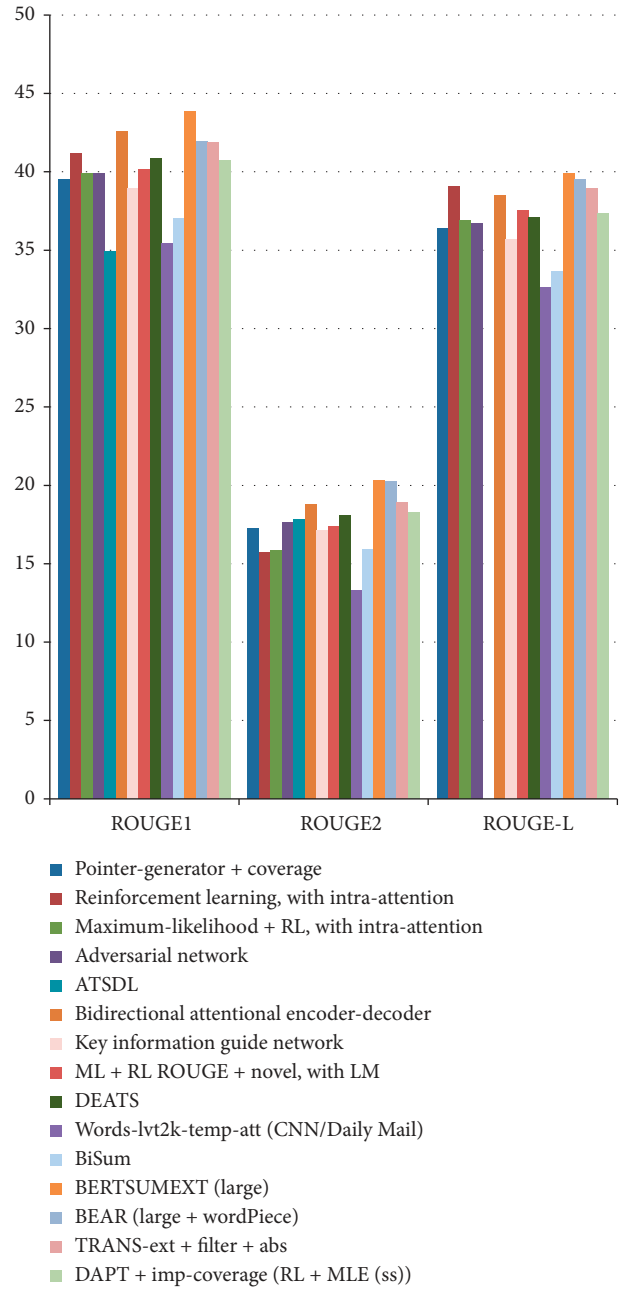

Figure 19: ROUGE1, ROUGE2, and ROUGE-L scores of abstractive text summarisation methods for the CNN/Daily Mail datasets.

unseen tokens to copy them. Moreover, in [30], rare words were addressed by using the location of the phrase, and the resulting summary was more natural. Moreover, in 35 [35, 58, 60], the OOV problem was addressed by using the pointer-generator technique employed in [56], which alternates between generating a new word and coping the word from the original input text.

7.3. Summary Sentence Repetition and Inaccurate Information Summary. The repetition of phrases and generation of incoherent phrases in the generated output summary are two challenges that must be considered. Both challenges are due to the summarisation of long documents and the production of long summaries using the attention-based encoder-decoder RNN [57]. In [35, 56], repetition was addressed by using the coverage model to create the coverage vector by aggregating the attention over all previous timesteps. In [57], repetition was addressed by using the key attention mechanism, where for each input token, the encoder intra-temporal attention records the weights of the previous attention. Furthermore, the intratemporal attention uses the hidden states of the decoder at a certain timestep, the previously generated words, and the specific part of the encoded input sequence, as shown in Figure 22, to prevent repetition and attend to the same sequence of the input at a different step of the decoder. However, the intra-attention encoder mechanism cannot address all the repetition challenges, especially when a long sequence is generated. Thus, the intradecoder attention mechanism was proposed to allow the decoder to consider more previously generated words. Moreover, the proposed intradecoder attention mechanism is applicable to any type of the RNN decoder. Repetition was also addressed by using an objective function that combines the cross-entropy loss maximum likelihood and gradient reinforcement learning to minimise the exposure bias. In addition, the probability of trigram $p\ (yt)$ was proposed to address repetition in the generated summary, where yt is the trigram sequence. In this case, the value of $p\ (yt)$ is 0 during a beam search in the decoder when the same trigram sequence was already generated in the output summary. Furthermore, in [60], the heuristic proposed by [57] was employed to reduce repetition in the summary. Moreover, in [61], the proposed approach addressed repetition by exploiting the encoding features generated using a secondary encoder to remember the previously generated decoder output, and the coverage mechanism is utilised.

7.4. Fake Facts. Abstractive summarisation may generate summaries with fake facts, and 30% of summaries generated from abstractive text summarisation suffer from this problem [53]. With fake facts, there may be a mismatch between the subject and the object of the predicates. Thus, to address this problem, dependency parsing and open information extraction (e.g., open information extraction (OpenIE)) are performed to extract facts.

Therefore, the sequence-to-sequence framework with dual attention was proposed, where the generated summary was conditioned by the input text and description of the extracted facts. OpenIE facilitates entity extraction from a relation, and Stanford CoreNLP was employed to provide the proposed approach with OpenIE and the dependency parser. Moreover, the decoder utilised copying and coverage mechanisms.

7.5. Other Challenges. The main issue of the abstractive text summarisation dataset is the quality of the reference summary (Golden summary). In the CNN/Daily Mail dataset, the reference summary is the highlight of the news. Every highlight represents a sentence in the summary; therefore,

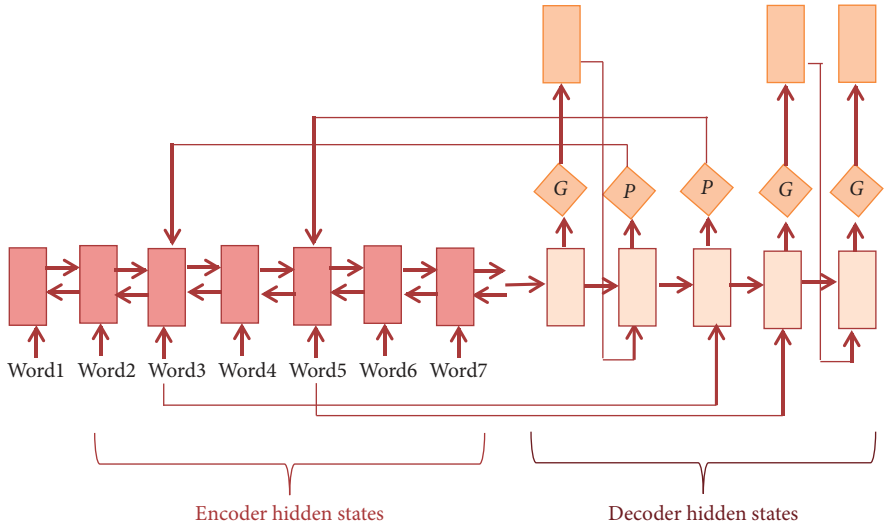

Figure 20: The generator/pointer switching model [55].

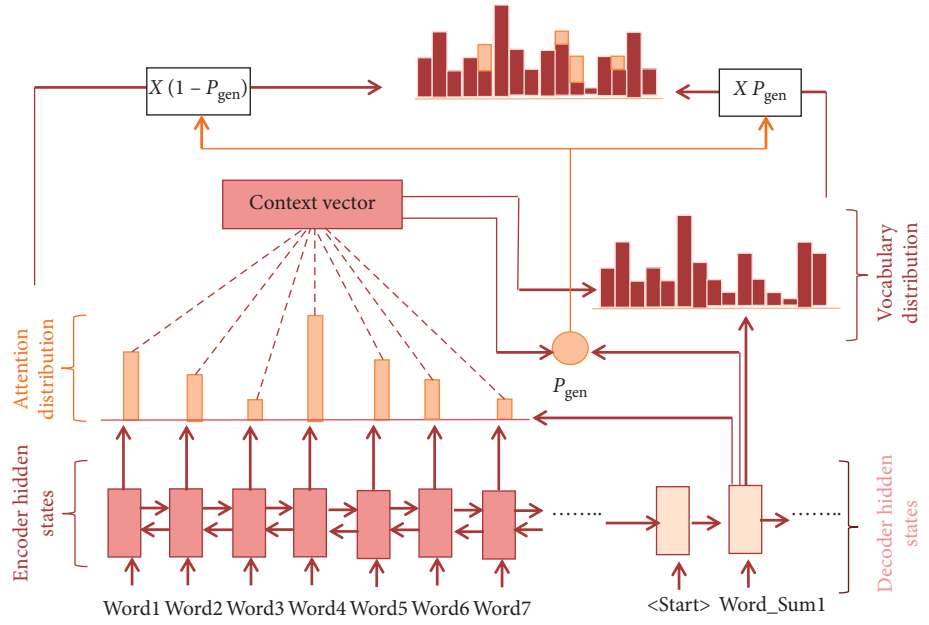

Figure 21: Pointer-generator model [56].

the number of sentences in the summary is equal to the number of highlights. Sometimes, the highlights do not address all crucial points in the summary. Therefore, a high-quality dataset needs high effort to become available. Moreover, in some languages, such as Arabic, the multi-sentence dataset for abstractive summarisation is not available. Single-sentence abstractive Arabic text summarisation is available but is not free.

Another issue of abstractive summarisation is the use of ROUGE for evaluation. ROUGE provides reasonable results in the case of extractive summarisation. However, in abstractive summarisation, ROUGE is not enough as ROUGE depends on exact matching between words. For example, the words book and books are considered different using any one of the ROUGE metrics. Therefore, a new evaluation measure must be proposed to consider the context of the words (words that have the same meaning must be considered the same even if they have a different surface form). In this case, we propose to use METEOR which was used recently in evaluating machine translation and automatic summarisation models [77]. Moreover, METEOR considers stemming, morphological variants, and synonyms. In addition, in flexible order language, it is better to use ROUGE without caring about the order of the words.

The quality of the generated summary can be improved using linguistic features. For example, we proposed the use

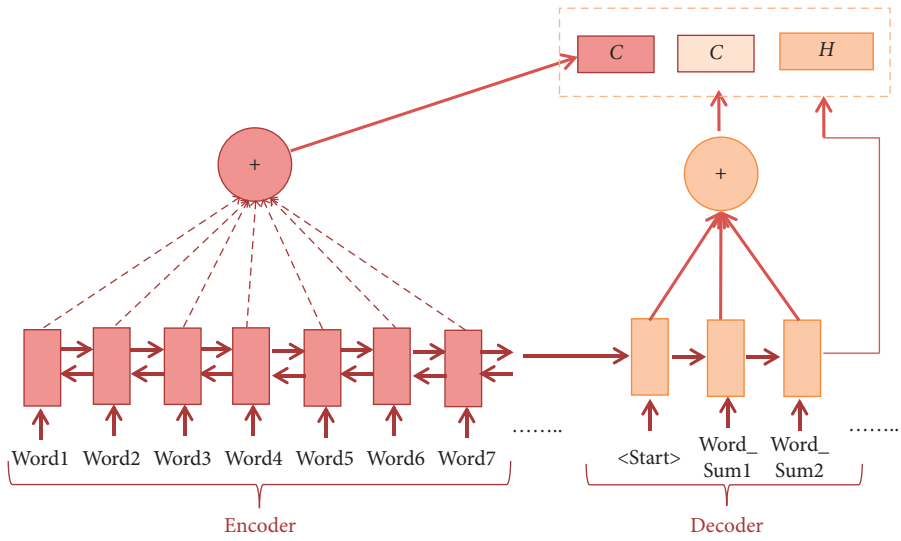

Figure 22: A new word is added to the output sequence by combining the current hidden state "H" of the decoder and the two context vectors, marked as "C" [57].

of dependency parsing at the encoder in a separate layer at the top of the first hidden state layer. We proposed the use of the word embedding, which was built by considering the dependency parsing or part-of-speech tagging. At the decoder side, the beam-search quality can be improved by considering the part-of-speech tagging of the words and its surrounding words.

Based on the new trends and evaluation results, we think that the most promising feature among all the features is the use of the BERT pretrained model. The quality of the models that are based on the transformer is high and will yield promising results.

8. Conclusion and Discussion

In recent years, due to the vast quantity of data available on the Internet, the importance of the text summarisation process has increased. Text summarisation can be divided into extractive and abstractive methods. An extractive text summarisation method generates a summary that consists of words and phrases from the original text based on linguistics and statistical features, while an abstractive text summarisation method rephrases the original text to generate a summary that consists of novel phrases. This paper reviewed recent approaches that applied deep learning for abstractive text summarisation, datasets, and measures for evaluation of these approaches. Moreover, the challenges encountered when employing various approaches and their solutions were discussed and analysed. The overview of the reviewed approaches yielded several conclusions. The RNN and attention mechanism were the most commonly employed deep learning techniques. Some approaches applied LSTM to solve the gradient vanishing problem that was encountered when using an RNN, while other approaches applied a GRU. Additionally, the sequence-to-sequence model was utilised for abstractive summarisation. Several datasets were employed, including Gigaword, CNN/Daily Mail, and the New York Times. Gigaword was selected for single-sentence summarisation, and CNN/Daily Mail was employed for multisentence summarisation. Furthermore, ROUGE1, ROUGE2, and ROUGE-L were utilised to evaluate the quality of the summaries. The experiments showed that the highest values of ROUGE1, ROUGE2, and ROUGE-L were obtained in text summarisation with a pretrained encoder mode, with values of 43.85, 20.34, and 39.9, respectively. The best results were achieved by the models that apply Transformer. The most common challenges faced during the summarisation process were the unavailability of a golden token at testing time, the presence of OOV words, summary sentence repetition, sentence inaccuracy, and the presence of fake facts. In addition, there are several issues that must be considered in abstractive summarisation, including the dataset, evaluation measures, and quality of the generated summary.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] M. Allahyari, S. Pouriyeh, M. Assefi et al., "Text summarization techniques: a brief survey," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 10, 2017.

[2] A. B. Al-Saleh and M. E. B. Menai, "Automatic Arabic text summarization: a survey," *Artificial Intelligence Review*, vol. 45, no. 2, pp. 203–234, 2016.

[3] A. Turpin, Y. Tsegay, D. Hawking, and H. E. Williams, "Fast generation of result snippets in web search," in *Proceedings of the 30th Annual international ACM SIGIR Conference on*

Research and Development in information Retrieval-SIGIR'07, p. 127, Amsterdam, The Netherlands, 2007.

[4] E. D. Trippe, "A vision for health informatics: introducing the SKED framework an extensible architecture for scientific knowledge extraction from data," 2017, http://arxiv.org/abs/1706.07992.

[5] S. Syed, *Abstractive Summarization of Social Media Posts: A case Study using Deep Learning*, Master's thesis, Bauhaus University, Weimar, Germany, 2017.

[6] D. Suleiman and A. A. Awajan, "Deep learning based extractive text summarization: approaches, datasets and evaluation measures," in *Proceedings of the 2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pp. 204–210, Granada, Spain, 2019.

[7] Q. A. Al-Radaideh and D. Q. Bataineh, "A hybrid approach for Arabic text summarization using domain knowledge and genetic algorithms," *Cognitive Computation*, vol. 10, no. 4, pp. 651–669, 2018.

[8] C. Sunitha, A. Jaya, and A. Ganesh, "A study on abstractive summarization techniques in Indian languages," *Procedia Computer Science*, vol. 87, pp. 25–31, 2016.

[9] D. R. Radev, E. Hovy, and K. McKeown, "Introduction to the special issue on summarization," *Computational Linguistics*, vol. 28, no. 4, pp. 399–408, 2002.

[10] A. Khan and N. Salim, "A review on abstractive summarization methods," *Journal of Theoretical and Applied Information Technology*, vol. 59, no. 1, pp. 64–72, 2014.

[11] N. Moratanch and S. Chitrakala, "A survey on abstractive text summarization," in *Proceedings of the 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pp. 1–7, Nagercoil, India, 2016.

[12] S. Shimpikar and S. Govilkar, "A survey of text summarization techniques for Indian regional languages," *International Journal of Computer Applications*, vol. 165, no. 11, pp. 29–33, 2017.

[13] N. R. Kasture, N. Yargal, N. N. Singh, N. Kulkarni, and V. Mathur, "A survey on methods of abstractive text summarization," *International Journal for Research in Emerging Science andTechnology*, vol. 1, no. 6, p. 5, 2014.

[14] P. Kartheek Rachabathuni, "A survey on abstractive summarization techniques," in *Proceedings of the 2017 International Conference on Inventive Computing and Informatics (ICICI)*, pp. 762–765, Coimbatore, 2017.

[15] S. Yeasmin, P. B. Tumpa, A. M. Nitu, E. Ali, and M. I. Afjal, "Study of abstractive text summarization techniques," *American Journal of Engineering Research*, vol. 8, 2017.

[16] A. Khan, N. Salim, H. Farman et al., "Abstractive text summarization based on improved semantic graph approach," *International Journal of Parallel Programming*, vol. 46, no. 5, pp. 992–1016, 2018.

[17] Y. Jaafar and K. Bouzoubaa, "Towards a new hybrid approach for abstractive summarization," *Procedia Computer Science*, vol. 142, pp. 286–293, 2018.

[18] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, 2015.

[19] N. Raphal, H. Duwarah, and P. Daniel, "Survey on abstractive text summarization," in *Proceedings of the 2018 International Conference on Communication and Signal Processing (ICCSP)*, pp. 513–517, Chennai, 2018.

[20] Y. Dong, "A survey on neural network-based summarization methods," 2018, http://arxiv.org/abs/1804.04589.

[21] A. Mahajani, V. Pandya, I. Maria, and D. Sharma, "A comprehensive survey on extractive and abstractive techniques for text summarization," in *Ambient Communications and Computer Systems*, Y.-C. Hu, S. Tiwari, K. K. Mishra, and M. C. Trivedi, Eds., vol. 904, pp. 339–351, Springer, Singapore, 2019.

[22] T. Shi, Y. Keneshloo, N. Ramakrishnan, and C. K. Reddy, *Neural Abstractive Text Summarization with Sequence-To-Sequence Models: A Survey*, http://arxiv.org/abs/1812.02303, 2020.

[23] A. Joshi, E. Fidalgo, E. Alegre, and U. de León, "Deep learning based text summarization: approaches, databases and evaluation measures," in *Proceedings of the International Conference of Applications of Intelligent Systems*, Spain, 2018.

[24] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[25] D. Suleiman, A. Awajan, and W. Al Etaiwi, "The use of hidden Markov model in natural Arabic language processing: a survey," *Procedia Computer Science*, vol. 113, pp. 240–247, 2017.

[26] H. Wang and D. Zeng, "Fusing logical relationship information of text in neural network for text classification," *Mathematical Problems in Engineering*, vol. 2020, pp. 1–16, 2020.

[27] J. Yi, Y. Zhang, X. Zhao, and J. Wan, "A novel text clustering approach using deep-learning vocabulary network," *Mathematical Problems in Engineering*, vol. 2017, pp. 1–13, 2017.

[28] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing [review article]," *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[29] K. Lopyrev, *Generating news headlines with recurrent neural networks*, p. 9, 2015, https://arxiv.org/abs/1512.01712.

[30] S. Song, H. Huang, and T. Ruan, "Abstractive text summarization using LSTM-CNN Based Deep Learning," *Multimedia Tools and Applications*, 2018.

[31] C. L. Giles, G. M. Kuhn, and R. J. Williams, "Dynamic recurrent neural networks: theory and applications," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 153–156, 1994.

[32] A. J. Robinson, "An application of recurrent nets to phone probability estimation," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 298–305, 1994.

[33] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the International Conference on Learning Representations*, Canada, 2014, http://arxiv.org/abs/1409.0473.

[34] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[35] K. Al-Sabahi, Z. Zuping, and Y. Kang, *Bidirectional Attentional Encoder-Decoder Model and Bidirectional Beam Search for Abstractive Summarization*, Cornell University, Ithaca, NY, USA, 2018, http://arxiv.org/abs/1809.06662.

[36] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[37] K. Cho, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, 2014.

[38] E. Jobson and A. Gutiérrez, *Abstractive Text Summarization Using Attentive Sequence-To-Sequence RNNs*, p. 8, 2016.

[39] S. Chopra, M. Auli, and A. M. Rush, "Abstractive sentence summarization with attentive recurrent neural networks," in *Proceedings of the NAACL-HLT16*, pp. 93–98, San Diego, CA, USA, 2016.

[40] C. Sun, L. Lv, G. Tian, Q. Wang, X. Zhang, and L. Guo, "Leverage label and word embedding for semantic sparse web service discovery," *Mathematical Problems in Engineering*, vol. 2020, Article ID 5670215, 8 pages, 2020.

[41] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, http://arxiv.org/abs/1301.3781.

[42] D. Suleiman, A. Awajan, and N. Al-Madi, "Deep learning based technique for Plagiarism detection in Arabic texts," in *Proceedings of the 2017 International Conference on New Trends in Computing Sciences (ICTCS)*, pp. 216–222, Amman, Jordan, 2017.

[43] D. Suleiman and A. Awajan, "Comparative study of word embeddings models and their usage in Arabic language applications," in *Proceedings of the 2018 International Arab Conference on Information Technology (ACIT)*, pp. 1–7, Werdanye, Lebanon, 2018.

[44] J. Pennington, R. Socher, and C. Manning, "Glove: global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, Doha, Qatar, 2014.

[45] D. Suleiman and A. A. Awajan, "Using part of speech tagging for improving Word2vec model," in *Proceedings of the 2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*, pp. 1–7, Amman, Jordan, 2019.

[46] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, "FastText.zip: compressing text classification models," 2016, http://arxiv.org/abs/161203651.

[47] A. Vaswani, N. Shazeer, N. Parmar et al., "Attention is all you need," *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

[48] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Pretraining of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4171–4186, Minneapolis, MN, USA, 2019.

[49] Z. Li, Z. Peng, S. Tang, C. Zhang, and H. Ma, "Text summarization method based on double attention pointer network," *IEEE Access*, vol. 8, pp. 11279–11288, 2020.

[50] J. Bradbury, S. Merity, C. Xiong, and R. Socher, *Quasi-recurrent neural networks*, https://arxiv.org/abs/1611.01576, 2015.

[51] U. Khandelwal, P. Qi, and D. Jurafsky, *Neural Text Summarization*, Stanford University, Stanford, CA, USA, 2016.

[52] Q. Zhou, N. Yang, F. Wei, and M. Zhou, "Selective encoding for abstractive sentence summarization," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pp. 1095–1104, Vancouver, Canada, July 2017.

[53] Z. Cao, F. Wei, W. Li, and S. Li, "Faithful to the original: fact aware neural abstractive summarization," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, New Orleans, LA, USA, February 2018.

[54] T. Cai, M. Shen, H. Peng, L. Jiang, and Q. Dai, "Improving transformer with sequential context representations for abstractive text summarization," in *Natural Language Processing and Chinese Computing*, J. Tang, M.-Y. Kan, D. Zhao, S. Li, and H. Zan, Eds., pp. 512–524, Springer International Publishing, Cham, Switzerland, 2019.

[55] R. Nallapati, B. Zhou, C. N. dos Santos, C. Gulcehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence RNNs and beyond," in *Proceedings of the CoNLL-16*, Berlin, Germany, August 2016.

[56] A. See, P. J. Liu, and C. D. Manning, "Get to the point: summarization with pointer-generator networks," in *Proceedings of the 55th ACL*, pp. 1073–1083, Vancouver, Canada, 2017.

[57] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," 2017, http://arxiv.org/abs/1705.04304.

[58] K. S. Bose, R. H. Sarma, M. Yang, Q. Qu, J. Zhu, and H. Li, "Delineation of the intimate details of the backbone conformation of pyridine nucleotide coenzymes in aqueous solution," *Biochemical and Biophysical Research Communications*, vol. 66, no. 4, 1975.

[59] C. Li, W. Xu, S. Li, and S. Gao, "Guiding generation for abstractive text summarization based on key information guide network," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 55–60, New Orleans, LA, USA, 2018.

[60] W. Kryściński, R. Paulus, C. Xiong, and R. Socher, "Improving abstraction in text summarization," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Brussels, Belgium, November 2018.

[61] K. Yao, L. Zhang, D. Du, T. Luo, L. Tao, and Y. Wu, "Dual encoding for abstractive text summarization," *IEEE Transactions on Cybernetics*, pp. 1–12, 2018.

[62] X. Wan, C. Li, R. Wang, D. Xiao, and C. Shi, "Abstractive document summarization via bidirectional decoder," in *Advanced Data Mining and Applications*, G. Gan, B. Li, X. Li, and S. Wang, Eds., pp. 364–377, Springer International Publishing, Cham, Switzerland, 2018.

[63] Q. Wang, P. Liu, Z. Zhu, H. Yin, Q. Zhang, and L. Zhang, "A text abstraction summary model based on BERT word embedding and reinforcement learning," *Applied Sciences*, vol. 9, no. 21, p. 4701, 2019.

[64] E. Egonmwan and Y. Chali, "Transformer-based model for single documents neural summarization," in *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pp. 70–79, Hong Kong, 2019.

[65] Y. Liu and M. Lapata, "Text summarization with pretrained encoders," 2019, http://arxiv.org/abs/1908.08345.

[66] P. Doetsch, A. Zeyer, and H. Ney, "Bidirectional decoder networks for attention-based end-to-end offline handwriting recognition," in *Proceedings of the 2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 361–366, Shenzhen, China, 2016.

[67] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, Montreal, Quebec, Canada, December 2014.

[68] D. He, H. Lu, Y. Xia, T. Qin, L. Wang, and T.-Y. Liu, "Decoding with value networks for neural machine translation," in *Proceedings of the Advances in Neural Information Processing Systems*, Long Beach, CA, USA, December 2017.

[69] D. Harman and P. Over, "The effects of human variation in DUC summarization evaluation, text summarization branches out," *Proceedings of the ACL-04 Workshop*, vol. 8, 2004.

[70] C. Napoles, M. Gormley, and B. V. Durme, "Annotated Gigaword," in *Proceedings of the AKBC-WEKEX*, Montréal, Canada, 2012.

[71] K. M. Hermann, T. Kocisky, E. Grefenstette et al., "Machines to read and comprehend," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, Montreal, Quebec, Canada, December 2015.

[72] M. Grusky, M. Naaman, and Y. Artzi, "Newsroom: a dataset of 1.3 million summaries with diverse extractive strategies," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, New Orleans, LA, USA, pp. 708–719, June 2018.

[73] C.-Y. Lin, "ROUGE: a package for automatic evaluation of summaries," in *Proceedings of the 2004 ACL Workshop*, Barcelona, Spain, July 2004.

[74] A. Venkatraman, M. Hebert, and J. A. Bagnell, "Improving multi-step prediction of learned time series models," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 3024–3030, Austin, TX, USA, 2015.

[75] I. Goodfellow, A. Courville, and Y. Bengio, *Deep Learning*, MIT Press, Cambridge, MA, USA, 2015.

[76] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Proceedings of the Annual Conference on Neural Information Processing Systems*, pp. 1171–1179, Montreal, Quebec, Canada, December 2015.

[77] A. Lavie and M. J. Denkowski, "The Meteor metric for automatic evaluation of machine translation," *Machine Translation*, vol. 23, no. 2-3, pp. 105–115, 2009.