IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 17, 2020, accepted January 3, 2021, date of publication January 19, 2021, date of current version January 25, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3052783

A Survey of the State-of-the-Art Models in Neural Abstractive Text Summarization

AYESHA AYUB SYED[1], FORD LUMBAN GAOL[1], (Senior Member, IEEE), AND TOKURO MATSUO[2,3], (Member, IEEE)
[1]Department of Doctor of Computer Science, Bina Nusantara University, Jakarta 11480, Indonesia
[2]Graduate School of Industrial Technology, Advanced Institute of Industrial Technology, Tokyo 140-0011, Japan
[3]Department of M-Commerce and Multimedia Applications, Asia University, Taichung 41354, Taiwan

Corresponding author: Ayesha Ayub Syed (ayeshaayubsyed@yahoo.com)

ABSTRACT Dealing with vast amounts of textual data requires the use of efficient systems. Automatic summarization systems are capable of addressing this issue. Therefore, it becomes highly essential to work on the design of existing automatic summarization systems and innovate them to make them capable of meeting the demands of continuously increasing data, based on user needs. This study tends to survey the scientific literature to obtain information and knowledge about the recent research in automatic text summarization specifically abstractive summarization based on neural networks. A review of various neural networks based abstractive summarization models have been presented. The proposed conceptual framework includes five key elements identified as encoder-decoder architecture, mechanisms, training strategies and optimization algorithms, dataset, and evaluation metric. A description of these elements is also included in this article. The purpose of this research is to provide an overall understanding and familiarity with the elements of recent neural networks based abstractive text summarization models with an up-to-date review as well as to render an awareness of the challenges and issues with these systems. Analysis has been performed qualitatively with the help of a concept matrix indicating common trends in the design of recent neural abstractive summarization systems. Models employing a transformer-based encoder-decoder architecture are found to be the new state-of-the-art. Based on the knowledge acquired from the survey, this article suggests the use of pre-trained language models in complement with neural network architecture for abstractive summarization task.

INDEX TERMS Abstractive text summarization, encoder, decoder, training, optimization, evaluation, attention, transformer.

I. INTRODUCTION

In the present technological era, there is a significant increase in textual data in digital form and it is continuously multiplying. Automatic summarization systems provide convenience to deal with lengthy text data effectively in a time-efficient way. These systems attempt to produce summaries that are comprehensive, concise, fluent, and at the same time capable of retaining all salient information contained in a topic. Some of the applications of text summarization include search engine snippets generated as a result of searching a document as well as news websites that generate condensed news in the form of headlines to facilitate browsing [1]. Other applications include lawsuit abstraction as well as biomedical and clinical text summarization [2]. Several approaches have been taken to summarize text documents in the scientific literature. Automatic summarization systems can be modeled in two ways, either using an extractive approach or an abstractive approach. When modeled using extractive techniques, the main sections of the text are extracted based on some scoring criteria and then concatenated to produce the summary. Abstractive techniques are a bit more complex and challenging to work with because the text is paraphrased to generate a summary having words different from the original text. All summarization methods and models, whether extractive or abstractive, share the common purpose of generating summaries that are fluent, non-redundant, and coherent. Both approaches can be employed to generate summaries across either a single source document or multiple source documents. In the case of one source document, the summarization system produces a short outline of the document. When a summary is generated across multiple documents,

The associate editor coordinating the review of this manuscript and approving it for publication was Xin Luo.

readers can familiarize themselves with information through several documents related to the same topic in relatively less time.

The research in Automatic Text Summarization is enriched with many surveys that have been conducted and published in the past years. The survey conducted by [3] and [4] are quite extensive while those conducted by [1] and [5]–[7] are centered towards extractive and abstractive summarization. This research study tends to survey the scientific literature to obtain information and knowledge about the recent research in automatic text summarization specifically abstractive summarization based on neural networks to have an understanding and familiarity with the design of state-of-the-art models in this realm. The novelty of this work as compared to previous surveys is that in addition to a review of various models, it provides its audience with the complete picture of neural networks based abstractive summarization systems. Secondly, it presents material on the very recently released models BERT [8], GPT [9], BART [10]. Finally, it includes an analysis section that is unique to it as compared with other surveys. Some of the papers with models proposed for machine translation are included as a part of this survey because later on these models were utilized by abstractive summarization systems.

Section II of this article is an attempt to provide readers with a comprehensive background on research in automatic summarization systems followed by classical abstractive summarization approaches and then the deep learning-based concepts which are applied in the recent design of abstractive models. Section III of this article highlights some of the recent and advanced models in the area of abstractive text summarization. Section IV presents the Methodology of the undertaken research study as well as the research framework based on which selected papers from the relevant scientific literature are reviewed. Section IV also contains a brief description of the elements of the research framework. Section V is the analysis and results section followed by Section VI which concludes this article and also provides some suggestions for future work.

II. BACKGROUND

Some of the very early approaches to automatic text summarization were the use of statistical models with an ability to select and copy the most important words from the text but these models were not capable of generating new text or paraphrasing text because these models were not capable of understanding the context or the meaning of these words [11]. The work of Luhn [12] for summarization of scientific articles which was published in the IBM journal in 1958, makes use of statistical information gained from the frequency and distribution of words in a text to calculate the relative significance. The method was first applied to words and then to sentences. The sentences with the highest significance were extracted to create an automatic summary. Some researchers applied the cue method, title method, and location method to determine the weight of the sentence [1]. Prior research on automatic summarization systems highlighted certain challenges such as the need for intelligent systems that can analyze a language and understand its semantics at a deeper level as well as generate purposeful sentences/descriptions from input data like human language [3]. Most of the earlier research focussed on extractive summarization and then the trends gradually shifted towards abstractive methods. In abstractive summarization, the text is usually interpreted through Natural Language Processing (NLP) techniques which help generate new text containing the most critical information from the original text. Three pipelined tasks are at the core of abstractive summarization approaches [5]: information extraction, content selection, and surface realization. Information extraction gets useful information from text using noun phrases or verb phrases. It can also extract important information by employing query-based extraction. Content selection works by selecting a subset of important phrases from the extracted text to include in the resulting summary. The surface realization task combines selected words or phrases in an ordered sequence by using grammatical rules and lexicons (vocabulary along with its related knowledge on linguistic significance and usage). Abstractive text summarization methods can be broadly classified into three domains: a structure-based approach, semantic-based approach, and deep learning-based approach [13]. The methods which are based on the structure approach work by encoding data from text documents based on some logical arrangements, for example, templates or other structures like trees, ontology, lead and body, rules (classes and lists), and graphs. Semantic-based methods work on identifying noun and verb phrases by applying linguistic/semantic illustration of a text document as an input to the natural language generation system. These systems include multimodal semantic-based techniques, information item-based methods, semantic text representation, and semantic graph methods [13], [14]. Recently, there are several advancements in text summarization using the concepts and methods of deep learning where sequence to sequence models are known to be the foundation of most of the recent studies [6].

A. ARTIFICIAL NEURAL NETWORKS

These networks contain neurons. A neuron is a computational unit. *S*everal layers of neurons make up a neural network as illustrated in Fig. 1. The first layer is the input layer and the final one is the output layer. In between, there are hidden layers. The data moves sequentially through the layers. Learning takes place in the hidden layers.

The nodes in each layer are connected to all the nodes in the next layer. The parameters that are associated with each of these connections are called weights. In Fig. 2, node j receives incoming signals from every node i in the preceding layer. Each input $\mathrm{x_i}$ is associated with a weight w_{ji}. The incoming signal to node j is the weighted sum of all incoming signals. The signal then goes through the activation function to produce the output signal h_j. The network can be mathematically

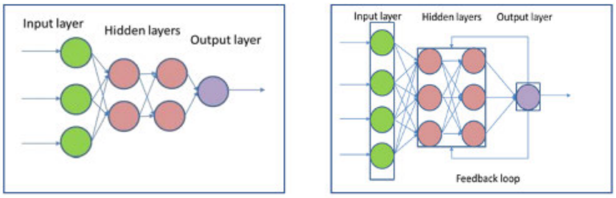


FIGURE 1. Feedforward and recurrent neural network.

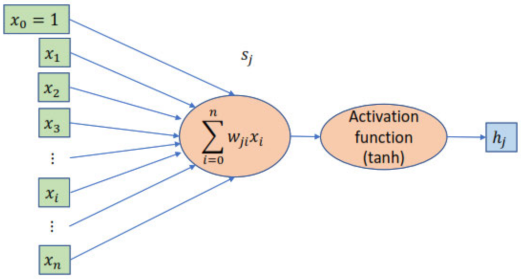


FIGURE 2. A node of the neural network.

expressed as,

$$s_j = \sum\nolimits_{i=0}^{n} w_{ji}x_i \tag{1}$$

$$h_j = \tanh s_j \tag{2}$$

$$h_j = 1 - \frac{2}{e^{2s_j} + 1} \tag{3}$$

B. CONVOLUTIONAL NEURAL NETWORK

The architecture of CNN is different from an ordinary neural network. The neurons in one layer do not have a connection to all neurons in the following layer but only to a specific part. However, there are fully connected layers near the end of the network. This type of network employs convolution units (where convolution operation is performed) in some (one or more) layers of the network, some pooling operations, and some of the neuron layers that are fully connected. A CNN has two parts, the feature extraction part (where convolutions and pooling operations are performed) and the classification part (fully connected layers near the output). Unlike simple neural networks, the layers are organized in the dimensions of weight, height, and depth. The output is a vector of probability scores. The work of [62] utilized CNN based deep learning approach for the task of abstractive text summarization. This approach was different from others as it searched important semantic phrases instead of sentences from the text and used these phrases to generate a summary.

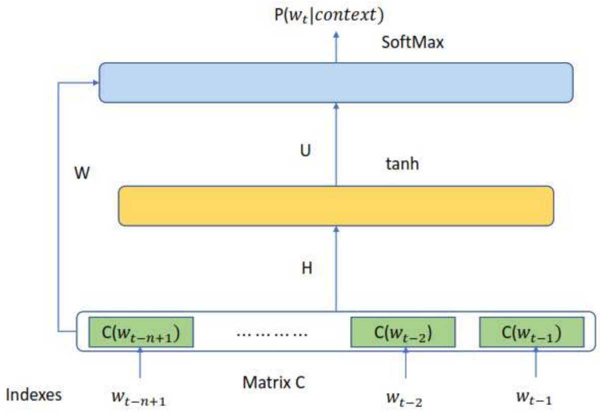
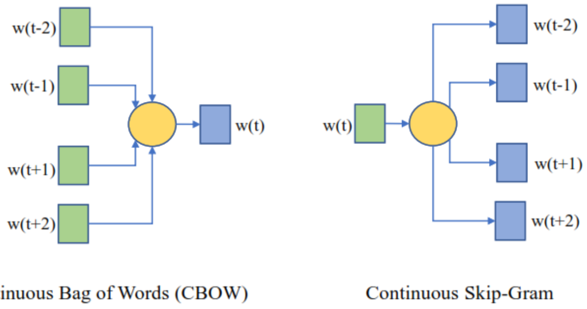


FIGURE 3. Neural network language model.



Continuous Bag of Words (CBOW)

Continuous Skip-Gram

FIGURE 4. CBOW and skip-gram models.

C. LANGUAGE MODELS

Language models are the foundation of many NLP tasks and applications that generate text as an output including text summarization, optical character recognition, part of speech tagging, parsing, handwritten text recognition (HTR), automatic language translation known as machine translation (MT), autocorrect for spelling, and generation of image caption [15]. In [16], the language model is defined as a mathematical function that places a probability measure over a list of strings from the lexicon of a particular language. These are systems that can analyze language text through an algorithm and learn to predict the words of a sentence by determining the probability of a sequence of words. In statistical language models for a sequence S having N-words, the probabilities are assigned as,

$$P(S) = P(w_1w_2w_3 \cdots w_N) \tag{4}$$

$$P(S) = P(w_1)\,P(w_2 \mid w_1) \cdots P(w_N \mid w_1w_2 \cdots w_{N-1}) \tag{5}$$

To reduce the parameters from the above equation, an approximate method was required. The n-gram model estimates the next word from the previous n-1 words.

$$P(w_N \mid w_1w_2 \cdots w_{N-1}) \approx P(w_N \mid w_{N-n} \cdots w_{N-1}) \tag{6}$$

On the other hand, to deal with the limitations of Statistical Language Models, Neural Network Language Models (NNLM) were introduced. With NNLM, it was possible to overcome the limitations of conventional language models and gain performance improvement [17]. These models can automatically learn features and representation and proved more effective in language modeling tasks. The first neural network language model was proposed by [18], as illustrated in Fig. 3.

The model in Fig. 3 can be expressed as,

$$y = b + Wx + U \tanh d + H(x) \tag{7}$$

where x is the feature vector formed by the concatenation of input word features, $x = \left(C(w_{t-1}), C(W_{t-2}), \cdots, C(w_{t-n+1})\right)$ and y is the probability of the output word. W, U, and H are weight matrices while b and d are the corresponding biases of the hidden and output layers.

Two models Continuous Bag-of-Words (CBOW) and Continuous skip-gram as shown in Fig. 4, were proposed by [19], for learning distributed word representations. The CBOW model learns by predicting a word from the context words (history and future), for example, a word w(t) is predicted from a shared projection of the input words w(t−1), w(t−2), w(t+1) and w(t+2). The skip-gram model learns representation by predicting a range of nearby words from a given word, for example, with w(t) as input to a log-linear classifier it is possible to predict w(t−1), w(t−2), w(t+1) and w(t+2).

Pre-trained language models utilize strategies for learning the parameters of neural network architectures and language representations that are universal as well as they can be fine-tuned on a variety of downstream tasks. Reference [63] classified these models as first and second-generation pre-trained language models. The first-generation pre-trained models are word embeddings (representation of words as vectors) and include word2vec, GloVe, etc., while the second-generation or more advanced models, also known as pre-trained contextual encoders (encoder output vector with context) include BERT [8], GPT [9], and ULMFiT (Universal Language Modeling fine Tuning), etc.

D. SEQUENCE TO SEQUENCE MODELS

These are the models where both input and output are sequences. Examples of sequence-to-sequence modeling problems include machine translation, image captioning, video captioning, speech recognition, language modeling, question answering system, text entailment, and text summarization. These models are encoder-decoder models. Each of the encoders and the decoder is an independent neural network. Both of these networks are combined to form a single large neural network. The encoder takes an input sequence and creates its representation. The decoder's function is to accept and decode the representation to generate another sequence as output.

III. RECENT MODELS IN ABSTRACTIVE TEXT SUMMARIZATION

This section describes some recently proposed models in the field of abstractive text summarization. It starts by mentioning the work of Facebook AI researchers [20] in which a model for sentence summarization is proposed. The model involves a neural network with attention. The researchers used a generation algorithm in conjunction with the model which helped the system to produce accurate summaries. Later, in another work [21], they updated their model with the conditional RNN [37] architecture. For conditioning, they introduced a convolutional attention-based encoder to help the decoder focus on pertinent input words at each step of summary formation. The new system outer-performed the previous work with the Gigaword corpus as well as performed very well on DUC 2004. The challenges highlighted in [20] are scaling the abstractive summarization system to generate paragraph-level summaries, efficient alignment, and consistency in the output generation.

In another study on abstractive summarization in Chinese language text, the researchers [22] first created a dataset called LCSTS (Large-scale Chinese Short Text Summarization) by using a microblogging website. This dataset forms a large corpus of Chinese brief texts with short outlines. This corpus was then utilized to introduce a summary generation model using RNNs. One of the suggestions for future work provided by the researchers is the use of hierarchical RNNs for summary generation and research towards rare word problems.

IBM's model [23] for abstractive text summarization is based on the neural encoder-decoder and incorporates attention. Word/morpheme/phrase embeddings are used as input. LSTM [40] and bidirectional RNNs [61] are used. Their work starts with the same framework as in [22] but incorporates other novel models that address critical problems in abstractive summarization. The basic model is the encoder-decoder model with attention and a large vocabulary trick. To address the challenge of capturing key concepts in a topic, a feature-rich encoder is used. To deal with OOV words, a switching generator/pointer mechanism is modeled. For long documents, in addition to capturing keywords, key sentences also need to be captured. This is achieved by using hierarchical attention.

Another work [24] is unique from the previous research because, in addition to using the attention mechanism to focus, the distraction mechanism introduced in it enabled traversing different parts of the document to grasp the overall meaning and create a better summary. The researchers used bidirectional gated recurrent units' architecture to perform encoding decoding tasks. The proposed work showed better performance than the work mentioned in [22] which the authors used for comparison.

The COPYNET model is proposed in [25] which incorporated the copying mechanism in the neural encoder-decoder model for sequence learning tasks. Replication is needed for those parts of the input sequence which do not need to be paraphrased like proper nouns and numerical data. The copying mechanism identifies a part of the input sequence and inserts it at the right position in the output sequence.

As reported by [26], already existing abstractive summarization systems have two problems, they produce factual details inaccurately and these systems sometimes produce a repetitive output. In this work, an abstractive summarization architecture that augments a hybrid pointer-generator and coverage with the traditional attention based seq2seq model is presented. The hybrid pointer-generator can copy words from the input text by pointing resulting in the correct reproduction of information while maintaining the ability to generate new words using the generator. This produces accurate information but it is still repetitive. Repetition is eliminated by employing a coverage mechanism that keeps a record of information that is already summarized. The model is evaluated on CNN/Daily Mail dataset [23] using ROUGE scores.

Google researchers [27] proposed a novel network architecture which they called the transformer. It was based entirely on attention mechanisms. Experiments on machine translation tasks indicated that it took significantly less time to train and the results were better as compared to other architectures. The transformer is the first sequence transduction model that can determine the input and output representations independent of using sequence recurrence or convolution.

Motivated by the bottom-up attention approach, the researchers at Harvard [28] applied it to neural abstractive summarization. Here, the content selector predetermines the phrases which should be part of the summary, and then it constrains the neural model by this content to produce an abstractive summary. This model shows higher performance and improvement of ROUGE scores on CNN/Daily mail [23] and NYT corpus as compared to [25] and [26].

Another hybrid architecture with an extractor module and an abstractor network abridged by policy-based reinforcement learning to address the issue of slow speed and inaccurate encoding of long documents is proposed by [29]. The model has achieved performance improvement on CNN/Daily mail corpus [23] using both ROUGE [48] and METEOR [50] scores. The extractor module is based on a convolutional model with a bidirectional LSTM-RNN at its output. Next, another LSTM-RNN is employed to train the pointer network. The abstractor module consists of a standard encoder-aligner-decoder framework with an added copy mechanism to deal with OOV words.

In [30], the researchers were keen to learn if knowledge from a pre-trained language model is helpful for the abstractive text summarization? So, they applied certain conditions to the encoder, decoder, and generator of the transformer-based neural model on the BERT (Bidirectional Encoder Representations from Transformer) [8] language model. They noted a big improvement with the encoder and decoder but not with the conditioning of the generator. The authors also introduced two-dimensional convolutional self-attention to the initial layers of the encoder. Then the models were compared with the most recent and advanced models on CNN/Daily Mail [23] datasets. The models were also trained on the German SwissText dataset to adapt the model to the German language also in addition to English. The BERT [8] based model showed better results than the self-attention convolutional model. Finally, to resolve the problem of long document summarization, TF-IDF (Term Frequency-Inverse Document Frequency) based extractive approach is applied and uses the BERT's next sentence prediction capability to increase the accuracy and reliability of the produced summaries. This approach made the system more efficient and consistent.

Reference [31] incorporates a contrastive attention mechanism into the model for the task of abstractive sentence summarization. The contrastive attention is different from traditional attention in the sense that it enhances the attention ability of the model by introducing double attention. One is the conventional attention that attends to the relevant parts of the text whereas the other is opponent attention that attends to the less relevant parts of a sentence. Both attentions are trained in an opposite manner using the softmax and the softmin functionality. The resulting attention performs better on the relevant parts of the input as compared to the ordinary attention and advances the best performance.

Most of the research in abstractive summarization targets to achieve a high ROUGE [48] score. The work of [2] addresses the factual inconsistencies of abstractive summarization systems and proposes a new evaluation metric to measure the factual correctness of abstractive summarization. The factual score computation task has three modules: fact extractor, fact encoder, and fact scorer. Facts are extracted from the generated summary and reference summary by using open information extraction (OpenIE) methods and then transformed into embeddings using the fact encoder. The fact scorer performs operations on each pair of generated summaries, reference summaries, and their embedding to compute the factual score. It has been concluded that the factual score has a high correlation with the BERT score and the ROUGE [48] score. The correlation is stronger in the case of BERT as compared to ROUGE.

The researchers [32] at Google recently proposed another abstractive summarization approach called PEGASUS (**P**re-training with **E**xtracted **G**ap-sentences **A**bstractive **Su**mmarization **S**equence to sequence models) which outer-performed previous models with even better results. The work is based on a standard transformer-based [27] encoder-decoder architecture. The authors first proposed a pre-training goal for multiple abstractive summarization tasks. They called this self-supervised objective GSG (Gap Sentences Generation). They identified principle sentence selection as their strategy. For pre-training, C4 (Colossal and Cleaned version of Common Crawl) and HugeNews datasets were used. For the downstream summarization task, they used XSum, Gigaword [20], arXiv, CNN/DailyMail [23], PubMed, Newsroom [47], Bigpatent, WikiHow, Reddit TIFU, Multi-News, AESLC, and BillSum. The task is evaluated using ROUGE scores.

Reference [11] applied deep neural network architecture for abstractive summarization to the Amharic language which is one of the African languages. As there was no already existing dataset available for the task, so first the researchers created an African language dataset from scratch. They used Google Colab as the training framework and results are evaluated using ROUGE [48] and BLEU [49] scores. Deep learning building blocks used in the system are sequence to sequence models using LSTM with attention, pointer generator network, and scheduled sampling approach.

The researchers at Microsoft's speech and dialogue research group [33] proposed an abstractive text summarization system for automatically generated meeting transcripts. According to them, meeting summarization is different from document summarization. Since many participants are engaged in a meeting, the nature of meeting transcripts is quite diverse due to varying semantic styles, viewpoints, and the roles of participants. The meeting summarization framework proposed is based on deep learning approaches and is named as Hierarchical Meeting summarization Network (HMNet). HMNet model makes use of the encoder-decoder transformer architecture [27] to produce abstractive summaries of the meeting transcripts. The model was evaluated on AMI and ICSI meeting corpus using three variations of the ROUGE [48] metric (Rouge-1, Rouge-2, Rouge-SU4).

An abstractive summarization model LPAS (Length-controllable Prototype-guided Abstractive Summarization) is proposed in [34] which can control the length of output summary. Before this research, it was common to use word embeddings to control the length of the summary. The idea of this research makes combined use of extractive and abstractive techniques. It accommodates a word-level extractive module in the seq2seq model. The design of the model has a prototype extractor, a joint encoder, and a summary decoder. The important words are extracted using the prototype extractor, important words, as well as the source text, are fed to the joint encoder and the abstractive summary is generated using the decoder. The authors used BERT [8] with the prototype extractor while the joint encoder-decoder used the transformer architecture [27]. The model was evaluated with CNN/Daily Mail [23] and Newsroom [47] datasets using ROUGE [48] scores.

The work of [35] is worth mentioning as it proposes a multilingual abstractive summarization model called Multi-Summ which is capable of dealing with numerous languages like English, Chinese, French, Spanish, German, Bosnian, and Croatian. The researchers implemented the training process in two stages, multilingual training (language model, auto-encoder model, translation and back translation model) and joint summary generation training. A new summarization dataset for the Bosnian and Croatian languages is also constructed. The model is implemented using transformer architecture [27]. BPE (Byte Pair Embedding) is used to process text in all languages. The datasets used are the Europarl-v5 dataset for English, German, Spanish and French, the News-Commentary-v13 dataset for Chinese and SETIMES dataset for Bosnian and Croatian.

The researchers [7] surveyed multiple aspects, methods, and components of neural Abstractive text summarization in detail and implemented the NATS (Neural Abstractive Text Summarization) toolkit. NATS is equipped with LSTM/GRU encoder-decoder [40], [42], and a pointer generator network. It is further enriched with mechanisms like intra-temporal attention, intra-decoder attention, coverage, weight sharing, beam search, and unknown words replacement technique. The model is evaluated on popular standard CNN/Daily Mail [23], Bytecup, and Newsroom [47] datasets using the ROUGE [48] metric.

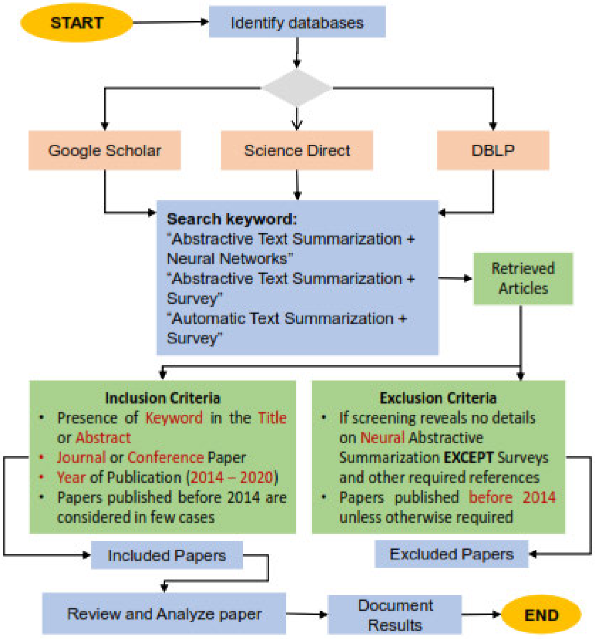

START
Identify databases
Google Scholar
Science Direct
DBLP
Search keyword:
"Abstractive Text Summarization + Neural Networks"
"Abstractive Text Summarization + Survey"
"Automatic Text Summarization + Survey"
Retrieved Articles
Inclusion Criteria
• Presence of Keyword in the Title or Abstract
• Journal or Conference Paper
• Year of Publication (2014 – 2020)
• Papers published before 2014 are considered in few cases
Exclusion Criteria
• If screening reveals no details on Neural Abstractive Summarization EXCEPT Surveys and other required references
• Papers published before 2014 unless otherwise required
Included Papers
Excluded Papers
Review and Analyze paper
Document Results
END

FIGURE 5. Research methodology.

An abstractive summarization system proposed by [36] uses Amazon fine food reviews dataset. Initially, data pre-processing is applied to the dataset which processes and transforms raw data to make it suitable for deep learning model/algorithm. This process consists of splitting text, adding contractions, removing stop words, and lemmatizing followed by vocabulary size counting, adding words embedding, and special tokens like UNK, EOS. The seq2seq model is built with a bidirectional [61] LSTM [40] encoder and a standard LSTM decoder. The limitation of the study as mentioned by the authors is that the accurate summary can be generated for short input text only. When lengthy text is entered as an input, the model becomes inconsistent.

IV. RESEARCH METHODOLOGY

This section describes the methodology by which the research is conducted. The research methodology is designed to meet the research objective of reviewing scientific literature regarding surveys, methods, and techniques of abstractive text summarization based on neural networks, to summarize some of the recent existing work in this area. As obvious from Fig. 5, the research process started with identifying the relevant databases for the collection of recent papers related to neural networks based abstractive text summarization. Some of the papers were collected using Google Scholar (scholar.google.com), a part of papers using Science Direct (www.sciencedirect.com), and still others using

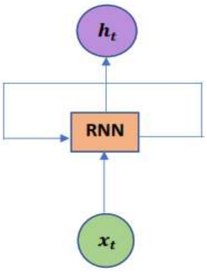


FIGURE 6. Recurrent neural network.

the DBLP (https://dblp.org/) database. Multiple search keywords were entered to retrieve the articles, e.g., ‘‘Abstractive Text Summarization + Neural Networks’’, Abstractive Text Summarization + Survey’’ and ‘‘Automatic Text Summarization + Survey’’. We defined inclusion and exclusion criteria for the retrieved articles as presented in Fig. 5. The final selection contains papers with keywords in the title or abstract which present novel work on abstractive summarization. We included papers that have been published either in conference proceedings or in journals. We emphasize selecting the recent papers (2014-2020) in the domain of neural networks based abstractive text summarization. But we have to include some papers published before the year 2014 because those papers provide the foundation of the recent research. Afterward, the selected papers were reviewed based on the proposed conceptual framework of the research topics as presented in Fig. 7. The papers were analyzed using the concept matrix. Finally, the results of the analysis are presented in the form of text, graphs, figures, and tables.

The research framework presented in Fig. 7 is created with the identification of the key elements and processes involved in the design of neural networks based abstractive summarization models. The main elements identified are *Encoder-Decoder Architecture*, *Mechanisms*, *Training* and *Optimization*, *Dataset,* and *Evaluation* metric. These elements have a further classification and sub-classification. The following paragraphs give a short introduction to these elements.

A. ENCODER DECODER ARCHITECTURE

The selection of encoder-decoder architecture provides us with certain choices of designing our encoder and decoder with standard RNN/LSTM/GRU, bidirectional RNN/LSTM/GRU, Transformer, BERT/GPT-2 architecture, or the very recent BART model.

1) RECURRENT NEURAL NETWORK (RNN)

RNN [37] has an architecture that is different from an ordinary/feedforward neural network. There are feedback loops that allow information to persist in these networks as indicated in Fig. 6. They introduce the concept of memory in neural networks. Due to their feedback nature, these networks can learn information based on the context.

In Fig. 6, the input of RNN is x_t and the previous hidden state h_{t-1}. Now, there is current input as well as information from the hidden state. Both combine as a vector and passes through tanh activation and produce the output h_t that becomes the memory of the network (new hidden state). Mathematically, RNN functionality can be expressed as,

$$h_t = \tanh(\omega_h h_{t-1} + \omega_x x_t) \tag{8}$$

Here, h_t is the current state, h_{t-1} is the previous hidden state and, ω_h is its weight. x_t is the current input, ω_x is the weight and, tanh is the activation function that determines the output of the neural network.

The architecture of RNN suits very well with tasks relating to sequential data. A novel network consisting of two RNNs as encoder and decoder was first proposed by [38] for statistical machine translation task. This model was further improved and extended by introducing a mechanism that automatically searches for parts of source text which shows relevance in predicting a target word (attention mechanism) for neural machine translation [39]. Although abstractive summarization is a unique function as compared to machine translation, yet they are closely related because both are sequence to sequence learning tasks. Many deep learning techniques are inspired by the success of neural machine translation and these are currently applied to generate abstractive summaries.

2) LONG SHORT-TERM MEMORY

LSTM introduced by [40], is a very special variant of RNN [37] because it can tackle the problem of long-term dependencies. For example, if the next word in a sequence is to be predicted and the correctly predicted word depends on information mentioned a few sentences before (past information), RNN is not capable of retaining information at length. This is where (long-term gaps/dependencies) LSTMs come into practice. LSTM can learn information for longer periods. Researchers [41] suggest that LSTM based approach can be successfully practiced on many sequence learning tasks with enough training data. Fig. 8 shows the architecture of an LSTM cell. There is a forget gate, an input gate, and an output gate in LSTM which work using the sigmoid activation function. The forget gate decides which information to keep or which information to forget. The input gate updates the state of the cell and the output gate decides what would be the next hidden state. The working of an LSTM cell is given by (9) to (14) as follows:

$$f_t = \sigma\left(\omega_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{9}$$

$$i_t = \sigma\left(\omega_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{10}$$

$$C'_t = \tanh\left(\omega_C \cdot [h_{t-1}, x_t] + b_C\right) \tag{11}$$

$$C_t = f_t * C_{t-1} + i_t * C'_t \tag{12}$$

$$o_t = \sigma(\omega_o [h_{t-1}, x_t] + b_o) \tag{13}$$

$$h_t = o_t * \tanh(C_t) \tag{14}$$

In (9) to (14) and Fig. 8, x_t is the input, C'_t is the candidate (holds possible values to add to the cell state), C_t is the new cell state, f_t is the output of forget gate, i_t is the output of the

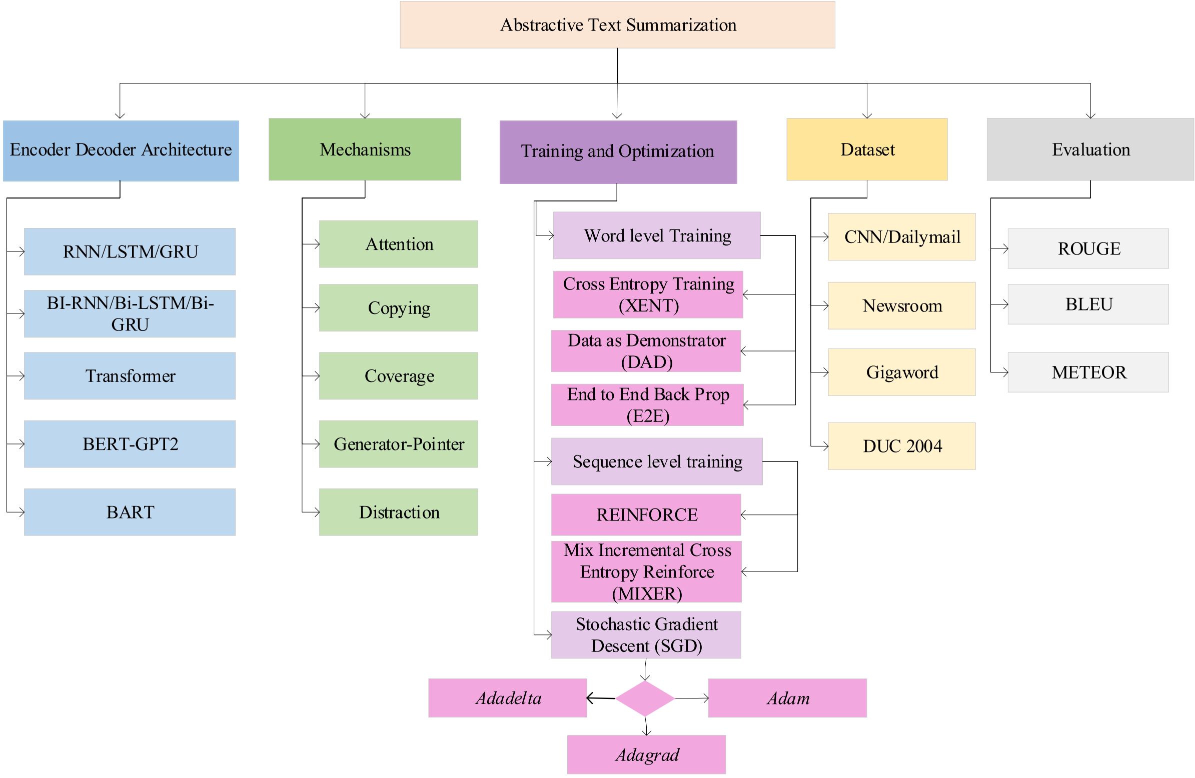


FIGURE 7. Abstractive text summarization research framework position.

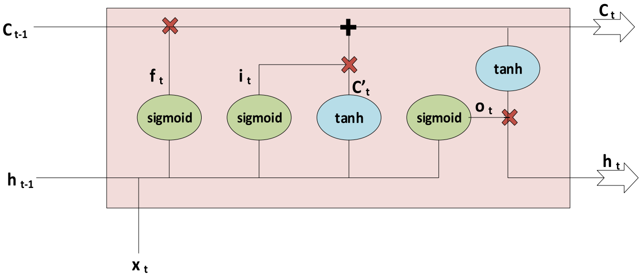


FIGURE 8. The architecture of an LSTM cell.

input gate, C_{t-1} is the previous state of the cell, h_{t-1} is the previous hidden state, h_t is the new hidden state while ω and b represent the corresponding weights and biases.

Vanishing Gradient is a very common problem in multi-layered neural networks. The more layers a neural network has, the more the capacity of the network which implies that it allows for better learning of large training datasets and is capable of mapping more complex functions from input to output. The problem arises when the gradient is backpropagated through the network, it gradually decreases in value. When it reaches near the initial layers, the gradient is considerably diminished. As a result, the weights and biases of the initial layers are not updated properly. Because the initial layers are very important in identifying the elements of input data, so the vanishing gradient leads to the imprecision of the whole network.

3) GATED RECURRENT UNIT (GRU)

GRU [42] is a variant of LSTM [40] because there is a commonality in the design of both. It addresses the problem of vanishing gradient in recurrent neural networks. The design of GRU has an update gate and a reset gate. The update gate deals with information that goes into the memory and helps the model to determine which of the past information needs to be passed on and the reset gate deals with information that flows out of the memory and helps the model to determine the past information which the network needs to forget. These are two vectors that decide about the information which is to be passed to the output. Fig. 9 presents the architecture and functionality of GRU. It can be expressed as follows:

$$z_t = \sigma\left(\omega_z \cdot [h_{t-1}, x_t]\right) \tag{15}$$

$$r_t = \sigma\left(\omega_r \cdot [h_{t-1}, x_t]\right) \tag{16}$$

$$h'_t = \tanh\left(\omega \cdot [r_t * h_{t-1}, x_t]\right) \tag{17}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * h'_t \tag{18}$$

In (15) to (18), x_t is the input, z_t is the update gate vector, r_t is the reset gate vector, h'_t is the tanh activation vector and h_t is the output. The ω, ω_r, and ω_z are the corresponding weight vectors.

4) BI-DIRECTIONAL RNN/LSTM/GRU

Bidirectional neural networks [61] consider two sequences for predicting the output, one in the forward direction and the other in the reverse direction. It implies that with bidirectional

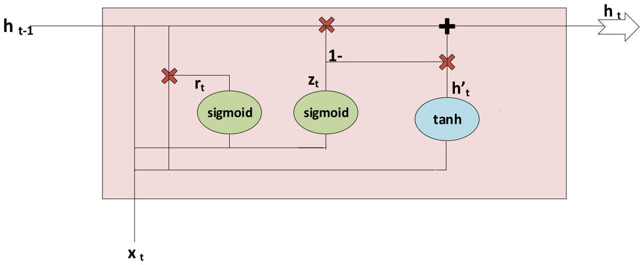


FIGURE 9. The architecture of gated recurrent unit (GRU).

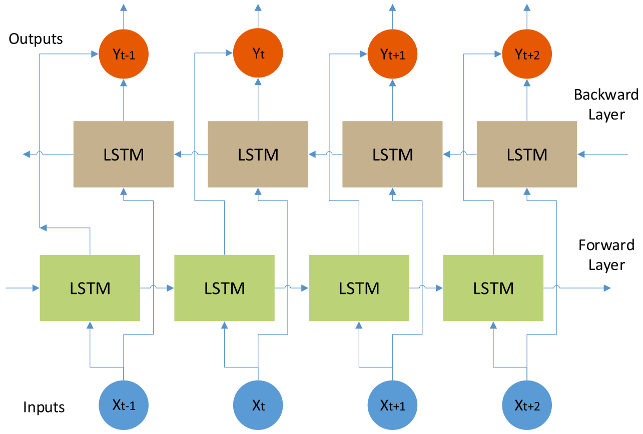


FIGURE 10. Bidirectional LSTM network.

networks we can make predictions of the current state by using information from previous time steps as well as later time steps. So, the network can capture a richer context and is capable of solving problems more effectively. Fig. 10 presents a bidirectional LSTM network. There are two LSTM layers: a forward layer and a backward layer. The input goes to the forward layer as well as to the backward layer. The output is a concatenation of the output of the forward and backward layers respectively.

5) TRANSFORMER

Transformers [27] are a breakthrough for sequence learning tasks introduced by Google in 2017. Transformers are based entirely on attention mechanisms thus eliminating the need for recurrent as well as convolution units. Its architecture has an encoder and a decoder that are stacked multiple times. The encoder and decoder blocks are made up of attention units and feed-forward units. The encoder part is a stack of six encoder units and the decoder part is a stack of six identical decoder units. Each encoder unit has a multi-head attention unit as well as a feedforward unit. Each decoder unit has an additional masked multi-head attention unit in addition to the feedforward unit and the multi-head attention unit. Fig. 11 presents transformer architecture. The functioning of the transformer starts with the word embeddings of the input sequence. The word embeddings are propagated to the first encoder which is then transformed and passed on to the following encoder. This process continues and the output of the last encoder unit is transferred to the decoder unit (all decoders in the stack).

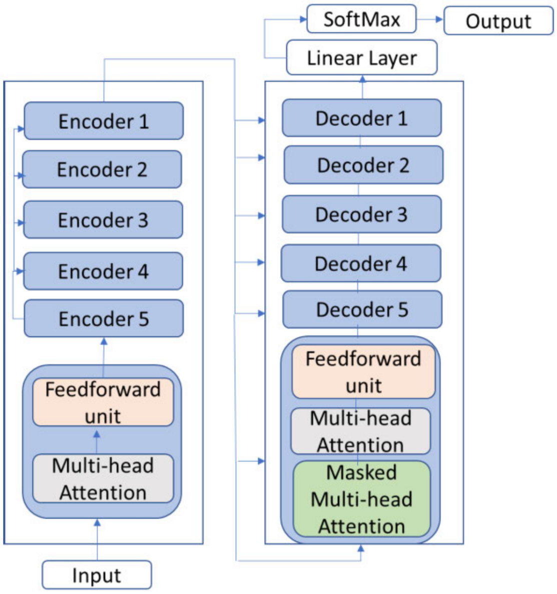


FIGURE 11. transformer architecture.

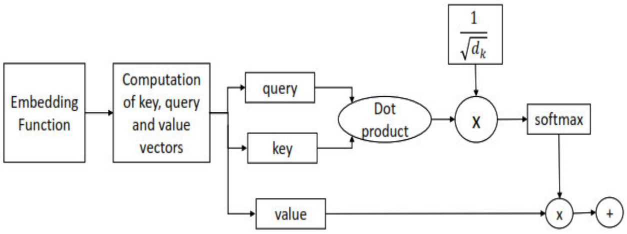


FIGURE 12. Attention mechanism in the transformer.

The attention mechanism in the transformer as shown in Fig. 12, is very interesting. The architecture is dependent on it for its functionality. First, three vectors called the key, query, and value are computed from the input embedding. There is a dot product operation between key and query vectors to calculate the attention weight. The attention score is scaled by $1/\sqrt{d_k}$ and passed on to the softmax function which is then multiplied with the value vector. This output is then passed to the feedforward layer. The self-attention in the transformer is not computed once but multiple times in parallel so it is also known as multi-head attention.

6) BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMER-GENERATIVE PRE-TRAINED TRANSFORMER (BERT-GPT)

BERT [8] introduced by Google in 2019 lets a pre-trained language model to be applied to a range of NLP tasks, while GPT [9] introduced by Open AI in 2018 also pre-trains a language model on a large body of text which can then be fine-tuned on a range of specific tasks. The key difference between the two is that BERT can perform bidirectional training whereas training is unidirectional in the case of

GPT. In terms of architecture, both are transformer-based. BERT is a multilayer transformer encoder while GPT is a multilayer transformer decoder. GPT-2 has an autoregressive nature, meaning that each token has a context of the previous words while BERT is not autoregressive and it employs all surrounding context at a time. Byte Pair Encodings (BPE) are used as word vectors for the first layer of GPT-2.

BERT has two modules pretraining and fine-tuning. BERT is trained with two tasks known as Masked Language Model (MLM) for bidirectional prediction and Next Sentence Prediction (NSP) for sentence-level understanding. For BERT, input embedding is the sum of token embeddings, segment embeddings, and position embeddings. In the MLM task, 15% of the words in the input sequence are replaced with a mask token and then the model predicts the masked word based on the context provided by other words in the sequence. A classification layer is added which receives the output of the encoder. Next, the output vectors are multiplied by the embedding matrix. Finally, each of the vocabulary words probability is computed using SoftMax. In the NSP task, a pair of sentences is given as input to the model, and the model functions to predict if the second sentence comes later in the original document. A [CLS] token is added at the start of the first sentence and a [SEP] token is put at the end of each sequence. A sentence identifying embedding is added to each token. A positional embedding is also added to every token to indicate its position in the sequence. By adding a layer to the core model BERT can be fine-tuned for many natural language tasks.

7) BIDIRECTIONAL AND AUTOREGRESSIVE TRANSFORMERS (BART)

BART very recently introduced by [10], has two major components, a bidirectional encoder, and an autoregressive decoder. Both components have transformer-based architecture and are implemented as a sequence-to-sequence model. While pre-training, a noising function introduces noise in the text and the system learns to reform the actual text from the distorted text. It is quite effective when fine-tuned for text generation and achieves leading results for many applications including abstractive dialogue, question answering, and text summarization [10]. The base model of BART uses 6 layers in each of the encoder and the decoder while the large model has 12 layers. While pretraining BART, several techniques are applied like token masking in which random tokens are replaced with [MASK]. In token deletion selected tokens are deleted and other tokens are inserted in their place. In text-infilling, some text areas are replaced by a [MASK] token. In sentence permutation, the sentences in the document are mingled up randomly. In document rotation, a random token is chosen to be the start of the document. The part of the document before the random token is inserted at the end. Fine-tuning BART allows the representations produced by it to be utilized by other applications such as sequence classification, token classification, sequence generation, and machine translation. The researchers further compared the pre-training objectives with those of other models like the GPT language model, Permuted Language model, masked language model, multitask masked language model, and masked sequence to sequence. The conclusions indicated token masking as very important, left-to-right pre-training improves Natural Language Generation (NLG) tasks and, bidirectionality is of key importance for question answering systems.

B. MECHANISMS

Mechanisms are functionalities added to the basic neural encoder-decoder architecture to address certain issues of abstractive summarization systems and to improve the generated summaries.

1) ATTENTION

The idea of the attention mechanism is to direct more focus and attention on certain chunks of input data as compared to others. It was introduced in [39] for neural machine translation. Later on, it was applied in many other tasks including abstractive summarization. When attention is applied the intermediate states of the encoder are utilized to create context vectors. When the system generates the output sequence, it searches for context vectors where the most relevant information is available. If attention is between input and output elements it is called general attention and if it is between the input elements then it is known as self-attention. Fig. 13 presents the attention model introduced by [39]. The model consists of a bidirectional RNN encoder and the RNN decoder.

As observed in Fig. 13, the encoder generates the hidden states h_1 to h_t. The context vector is computed as:

$$e_{ij} = a(s_{i-1}, h_j) \tag{19}$$

In (19), a is the alignment model.

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \tag{20}$$

The softMax function is used to normalize the alignment scores. The context vector is a weighted sum of α_{ij} and h_j.

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{21}$$

2) COPYING

When certain parts of the input sequence have to be replicated into the output sequence, the copying mechanism purposed in [25] can be used. This model is based on a bidirectional encoder that encodes the input sequence and a decoder that predicts the output sequence. A probabilistic model is used to predict the output sequence from the copy mode or the generate mode. Considering a set of vocabulary, $V = \{v_1, v_2, \ldots, v_n\}$, and an input sequence, $X = \{x_1, x_2, \ldots., x_{T_S}\}$. There might be words in the input sequence X, that are not in V. Here, the copy mode can copy words

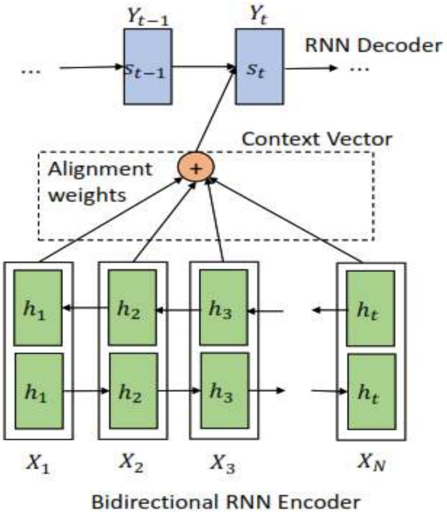


FIGURE 13. Attention mechanism proposed by [39].

from X that are not in V. If M is the representation of the input sequence from the encoder, s_t is the decoder state at time t and, c_t is the context, the probability of output word y_t is given by combined probabilities as

$$p\left(y_t \mid s_t, y_{t-1}, c_t, M\right) = p\left(y_t, g \mid s_t, y_{t-1}, c_t, M\right) + p\left(y_t, c \mid s_t, y_{t-1}, c_t, M\right) \quad (22)$$

where c and g are the copy and generate modes respectively.

3) COVERAGE

It was introduced to address the repetition problem in the output sequence and help eliminate or reduce it [26]. In this model, there is the tracking of the vocabulary that has already been covered. This is achieved by using attention distribution to keep the system aware of the covered sequence and the network is penalized in case of attending to the same sequence over again. Mathematically, at timestep t, the coverage vector c^t is represented as

$$c^t = \sum_{t'=0}^{t-1} \alpha^{t'} \quad (23)$$

The loss term that penalizes overlap between c^t and the new attention distribution α^t is expressed as

$$loss_t = \sum_i \min(\alpha_i^t, c_i^t) \quad (24)$$

4) POINTER-GENERATOR

This mechanism attempts to propose a method for eliminating the problem of out of vocabulary (OOV) words and factual details. It is capable of replicating words/facts via pointer while still maintaining the capacity to generate new words via a generator [26]. Together with computing an attention distribution α and a vocabulary distribution p_{vocab}, the model also calculates a generation probability denoted by p_{gen}. The generation probability represents the probability of predicting the final word either from the vocabulary or copying it from the input. Mathematically, the final probability of generating output word is expressed as,

$$p_{final}(w) = p_{gen} p_{vocab}(w) + (1 - p_{gen}) \sum_{i:w_i=w} \alpha_i \quad (25)$$

5) DISTRACTION

The distraction mechanism proposed by [24] is to distract in a way to traverse between different parts of a document to grasp the overall meaning for summarizing a document instead of focusing only on a specific part repeatedly. In this model, the researchers implemented distraction from a dual perspective, the first distraction task is performed in the training process and then in the decoding process. During training, the distraction was implemented on the content vector by training the model not to pay too much attention to the same part repeatedly. The already viewed vector was stored as a history content vector and merged with the currently computed vector.

Formally,

$$c_t = \tanh\left(W_c c'_t - U_c \sum_{j=1}^{t-1} c_j\right) \quad (26)$$

$$c'{}_t = \sum_{i=1}^{T_x} \alpha_{t,i} h_i \quad (27)$$

Where, c_j is the history content vector, c'_t is the input content vector, $\alpha_{t,i}$ is the attention weight at time t and h_i is the hidden state. Also, the distraction was put directly on the attention weight vectors. For this purpose, the previous attention weights were stored in a history attention weight vector and then cumulated with the currently computed attention weights.

$$a'_{t,i} = \upsilon_a^T \tanh\left(W_a s'_t + U_a h_i - b_a \sum_{j=1}^{t-1} \alpha_{j,i}\right) \quad (28)$$

And,

$$\alpha_{t,i} = \frac{\exp(\alpha'_{t,i})}{\sum_{j=1}^{T_x} \exp\left(\alpha'_{t,j}\right)} \quad (29)$$

Where, υ_a, W_a, U_a and, b_a are the weight matrices.

C. TRAINING AND OPTIMIZATION

Training is a process by which a model learns. The sequence-to-sequence models need to be trained to anticipate the following word in a sequence given the previous output and the context.

1) WORD LEVEL TRAINING

In word-level training, the language generation models can effectively improve the prediction of only one word in advance [43].

In Cross-Entropy Training (*XENT)*, the model is trained using the cross-entropy loss. Considering $[\mathrm{w}_1, \mathrm{w}_2, \mathrm{w}_3, \ldots \mathrm{w}_\mathrm{T}]$ as the target sequence, the purpose of XENT training is to minimize the loss as [43],

$$L = -\sum_{t=1}^{T} \log p(w_t | w_1, \ldots., w_{t-1}) \tag{30}$$

After training, the model generates the sequence as,

$$w_{t+1}^{g} = \arg\max_{w} p_\theta(w | w_t^g, h_{t+1}) \tag{31}$$

where, w_t^g is the word generated by the model at time step t and w_{t+1}^g is the next word that the model will generate. Now there is a problem associated with this approach, that, during training, the model is exposed to the ground truth words while during testing, the model takes into account its predictions. The result is that the generated sequence might be quite different from the one that should be generated. This is called search error and the process that can be used to reduce the effect of search error is known as beam search. In beam search, instead of predicting only one word, the strategy is to predict k next word candidates for each step. The strategy is quite effective with the downside that computation time increases with the number of beams.

Data as Demonstrator (DAD) proposed by [44] addresses the issue of exposure bias in XENT by combining ground truth data with model predictions. At each prediction point, the algorithm takes input either previous word from the model prediction or the ground truth data, based on a certain probability.

End-to-End Backprop (E2E) uses XENT in the initial steps and then uses a sparse vector with the largest probabilities of the distribution predicted at the previous time step. The idea is to take top scoring k previous words as input and then pass it through a k-max layer. This layer only keeps the k largest values, zeros out all the others, and renormalizes them to sum to 1. The model performs in a computationally efficient way to beam search [43].

2) SEQUENCE LEVEL TRAINING

In sequence-level training, the algorithm directly optimizes for final evaluation.

REINFORCE is an algorithm for sequence level training proposed by Williams in 1992. While using REINFORCE, the neural network is considered as an agent, the input words and the context vector are assumed as the external environment so that the neural network interacts with the external environment. Now, the parameters of the agent define a policy, that when executed causes the agent to choose an action. When the action is taken, the agent updates the internal state and on reaching the end of the sequence, the agent reaches a reward. Since we are discussing in the context of sequence generation, the action refers to predicting the next word in the sequence that results in an update of the states of the neural network. On reaching the end of the sequence, the reward function can be chosen as ROUGE, BLEU, or any other metric that is used to test the output. During training, the target is to find the agent parameters that result in a maximized reward [7], [43].

MIXER algorithm proposed by [43] is a combination of XENT and REINFORCE and uses incremental learning. MIXER works by changing the initial policy of REINFORCE suitably. It starts from an optimal policy and then slowly moves away from the optimal policy to use its predictions. The model is first trained with XENT loss for n epochs using ground truth data. This lets the model concentrate on the better part of the search space. Next, the model continues to train using XENT loss for the first few time steps and then using the REINFORCE for the remaining time steps. Then the length of time steps with XENT training is slowly reduced until only REINFORCE is used to train the entire sequence. The result of the whole process results in the effective use of model predictions during the training and testing time.

3) STOCHASTIC GRADIENT DESCENT (SGD)

SGD is a form of gradient descent which is one of the most common ways to do optimization of neural networks. Gradient descent minimizes an objective function $J(\theta)$, by moving in the opposite direction of the gradient of the objective function $\Delta_\theta J\ (\theta)$, and updating the parameters. The learning rate η is the size of the steps taken to reach the minimum value of the function. In SGD, replaces the actual gradient is replaced by a stochastic approximation computed from randomly selected data points instead of using all the data points in each iteration. For SGD, the parameter update is given as,

$$\theta = \theta - \eta . \Delta_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{32}$$

Adagrad (Adaptive gradient optimizer), Adadelta (Adaptive delta), and Adam (Adaptive Moment Estimation) are algorithms for gradient-based optimization [45]. In Adagrad optimizer, each parameter θ has a different learning rate at each time step t, which improves optimization. Mathematically,

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \varepsilon}} \cdot g_t \tag{33}$$

where G_t represents the sum of the squares of the past gradients and g_t is the gradient of the objective function at time t. Adagrad is extended to Adadelta which is more robust and it does not accumulate all past gradients instead it limits the window size to some fixed width. With Adadelta,

$$\theta_{t+1} = \theta_t + \Delta\theta_t \tag{34}$$

$$\Delta\theta = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g_t]} \cdot g_t \tag{35}$$

where RMS is the root mean squared error measure of the gradient. Adam is a widely used optimizer and is efficient with low memory requirements [46]. It works by updating the exponentially weighted averages of the first (gradient m_t)

and second moment (squared gradient v_t) estimates.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\, g_t \tag{36}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\, g_t^2 \tag{37}$$

where β_1 and β_2 are the hyperparameters that are to be tuned. The bias-corrected first-moment estimate (m_t') and second-moment estimate (v_t') are,

$$m_t' = \frac{m_t}{1-\beta_1^t} \tag{38}$$

$$v_t' = \frac{v_t}{1-\beta_2^t} \tag{39}$$

The parameter update occurs as,

$$\theta_{t+1} = \theta_t - \frac{\eta m'_t}{\sqrt{v'_t} + \varepsilon} \tag{40}$$

D. DATASETS

Datasets are useful for training and assessment of models. For abstractive text summarization, several datasets are available in the English language. A few of these datasets are mentioned in the research framework like the CNN/Daily Mail dataset which incorporates documents from news stories and editorials of CNN and Daily Mail. The dataset was introduced for abstractive summarization by [23] with the corpus containing 286,817 training pairs, 13,368 validation pairs, and 11,487 test pairs. Newsroom which is a summarization dataset of news publications was presented by [47]. It has 1.3 million articles and summaries written by authors from newsrooms of 38 news publications. The data timeline for the Newsroom dataset is between 1998 and 2017. Gigaword produced by LDC (Linguistic Data Consortium) is a comprehensive collection of Newswire documentation in English and was first employed for the task of abstractive summarization by [20] and it has approx. 9.5 million news articles. For this dataset, a source-summary pair was created from each article by using the first sentence of the article as the source and its headline as the summary [5]. The DUC (Document Understanding Conference) dataset for training and evaluating text summarization systems is available in two parts known as the 2003 corpus consisting of 624 document- summary pairs and the 2004 corpus consisting of 500 pairs [23].

E. EVALUATION METRIC

When the task of summarization is automated, it requires a system or method for its assessment and evaluation. Manual assessment is one way to evaluate automatically generated summaries. However, certain metrics also exist for this purpose. ROUGE (Recall Oriented Understudy for Gisting Evaluation) proposed by [48] is based on recall and most commonly employed in the evaluation of automatically generated summaries. BLEU (Bilingual Evaluation Understudy) proposed by [49] is based on precision and recall. BLEU scores are also used for the evaluation of automatic summarization systems [6]. METEOR (Metric for Evaluation of Translation with Explicit Ordering) proposed in [50] is basically for the interpretation of machine translation output but, it can be applied to summarization as well. METEOR is based on modified precision and recall. All the mentioned evaluation metrics provide an approximation of how much the automatically generated summary matches with the reference summary.

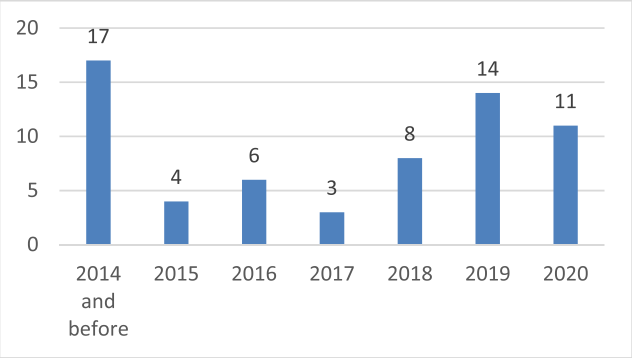


FIGURE 14. Bibliographic statistics of esearch publications in the survey.

TABLE 1. Statistics of encoder-decoder architecture element from the concept matrix.

Encoder-Decoder Architecture	No. of References
RNN/LSTM/GRU	6
Bidirectional RNN/LSTM/GRU	10
Transformer	6
BERT-GPT	3
BERT-Encoder Transformer Decoder	1
Other	1
Total	26

V. ANALYSIS, RESULTS & DISCUSSION

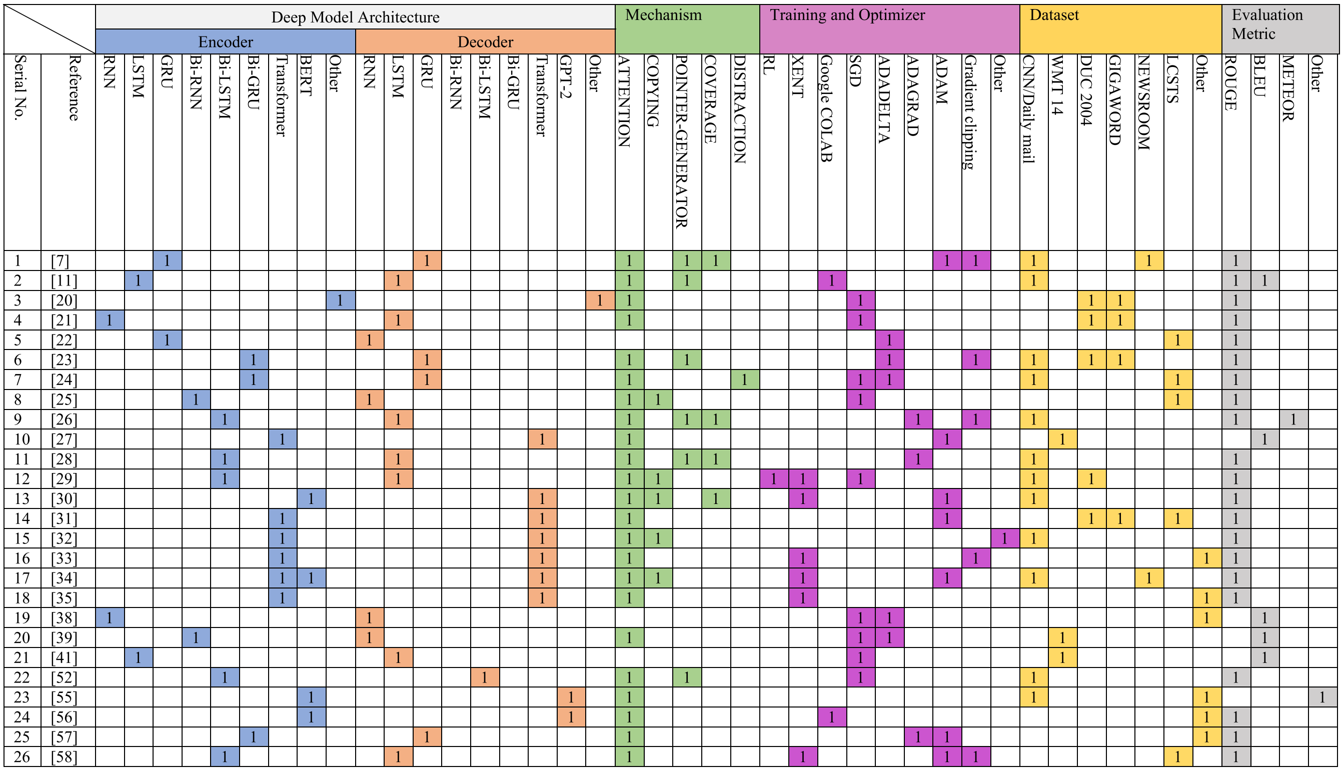
The contents of this section reveal the analysis and results of the survey as well as discuss important points. First of all, statistics of reference publications across the year of publication is presented in Fig. 14. There is a total of 63 papers that are distributed across the years 2014 (and before) to 2020. Almost 50% of the papers considered are the recent papers of the years 2018, 2019 and, 2020.

Webster and Watson [51] identified two approaches to literature reviews in their article. The first one is concept-centric, another one is author-centric. We have applied both approaches to this study. The approach adopted in section III is author-centric while the approach presented in Fig. 15 is concept-centric. A concept matrix is created in line with the research framework. Concepts/elements are entered across columns while references are presented across rows. 26 papers are included in the concept matrix.

A. ENCODER DECODER ARCHITECTURE

Table 1 summarizes the statistics of the encoder-decoder architecture element from the concept matrix shown in Fig. 15.

It can be observed from Table 1 that out of a total of 26 papers, 6 papers use an encoder-decoder architecture that employs standard unidirectional RNN/ LSTM/GRU as either encoder, decoder, or both. 9 papers



		Deep Model Architecture																		Mechanism					Training and Optimizer									Dataset							Evaluation Metric			
		Encoder									Decoder																																	
Serial No.	Reference	RNN	LSTM	GRU	Bi-RNN	Bi-LSTM	Bi-GRU	Transformer	BERT	Other	RNN	LSTM	GRU	Bi-RNN	Bi-LSTM	Bi-GRU	Transformer	GPT-2	Other	ATTENTION	COPYING	POINTER-GENERATOR	COVERAGE	DISTRACTION	RL	XENT	Google COLAB	SGD	ADADELTA	ADAGRAD	ADAM	Gradient clipping	Other	CNN/Daily mail	WMT 14	DUC 2004	GIGAWORD	NEWSROOM	LCSTS	Other	ROUGE	BLEU	METEOR	Other
1	[7]			1									1							1		1	1								1	1		1				1			1			
2	[11]		1									1								1		1					1							1							1	1		
3	[20]									1									1	1								1								1	1				1			
4	[21]	1										1								1								1								1	1				1			
5	[22]			1							1																		1										1		1			
6	[23]						1						1							1		1							1			1		1		1	1				1			
7	[24]						1						1							1				1				1	1					1					1		1			
8	[25]				1						1									1	1							1											1		1			
9	[26]					1						1								1		1	1							1		1		1							1		1	
10	[27]							1									1			1											1				1							1		
11	[28]					1						1								1		1	1							1				1							1			
12	[29]					1						1								1	1				1	1		1						1		1					1			
13	[30]								1								1			1	1		1			1					1			1							1			
14	[31]							1									1			1											1					1	1		1		1			
15	[32]							1									1			1	1												1	1							1			
16	[33]							1									1			1						1						1								1	1			
17	[34]							1	1								1			1	1					1					1			1				1			1			
18	[35]							1									1			1						1														1	1			
19	[38]	1									1																	1	1											1		1		
20	[39]				1						1									1								1	1						1							1		
21	[41]		1									1																1							1							1		
22	[52]					1									1					1		1						1						1							1			
23	[55]								1									1		1															1						1				1
24	[56]								1									1		1							1													1	1			
25	[57]						1						1							1										1	1									1	1			
26	[58]					1						1								1						1					1	1							1		1			

FIGURE 15. Concept matrix of the selected papers.

used an encoder based on bidirectional RNN/LSTM/GRU while none of their decoder parts is bidirectional instead it is based on standard RNN/LSTM/GRU. We have encountered only one paper [52] which is based on a bidirectional encoder as well as a bidirectional-decoder. 6 recent papers have transformer-based encoder and decoder design while 2 recent studies are based on BERT-GPT architecture and another one based on BERT encoder and standard transformer decoder.

Further investigation of recent research papers of 2019 and 2020 indicates the use of transformer-based architecture for modeling abstractive summarization systems. The reason being, the transformer is quite efficient during training because there are no recurrent neural network layers so no need to be trained with backpropagation through time. The experimental results of [53] indicate that the transformer performs better as compared to other sequence-to-sequence models on abstractive summarization tasks. Recently, pre-trained language models are successfully being employed in abstractive summarization systems to achieve better results. BART [10] and MASS (Masked Sequence to Sequence) [54] pre-training methods have achieved notable success for seq2seq language generation. During this survey, we have observed a lack of abstractive summarization models using BART and MASS architectures.

B. MECHANISMS

Table 2 summarizes the statistics of mechanisms employed by various researchers. It can be noted that all papers except a few incorporated some form of attention mechanism. Apart from attention, 5 papers included the copying functionality while another 6 included the pointer-generator model. 4 papers used the concept of coverage in their work while one paper introduced the distraction component.

TABLE 2. Statistics of mechanisms from the concept matrix.

Mechanism	No. of References
Attention	23
Copying	5
Pointer-generator	6
Coverage	4
Distraction	1

In addition to conventional attention, we have observed various techniques of attention implemented by different researchers which are listed in Table 3.

Here, we can conclude that the concept of attention is very important for abstractive summarization systems and considerable focus needs to be directed towards it while designing the models.

C. TRAINING AND OPTIMIZATION

For the training and optimization element of the research framework, it can be observed from Fig. 15 that the common training and optimization trend is by using cross-entropy loss and stochastic gradient descent. Reference [59] proposed training methods that are based on reinforcement learning. Reference [60] further extended the reinforcement learning concept for training with distributional semantic rewards and reported an overall improvement of the generated abstractive summary.

TABLE 3. Types of attention implemented by various models.

Attention Type	Reference
Hierarchical attention	[23]
Multi-headed self-attention	[27], [9],[34],[27]
Bottom-up attention	[28]
Contrastive attention	[31]
Intra-temporal attention	[7]
Intra-decoder attention	[7]
Cross attention	[10],[55]
Self-attention	[10]
Two stream attention	[10]
Bidirectional attention	[9]
Bidirectional self-attention	[8]
Bidirectional cross attention	[8]
Summary aware attention	[58]
Masked self-attention	[56]
Convolutional self-attention	[30]
2-hop attention	[29]
Scaled dot product attention	[27]

TABLE 4. Languages across the survey.

Language	Reference
English	[23],[24],[26],[28],[29],[30],[32],[34],[7],[55]
German	[30]
Chinese	[22],[24],[25],[35],[58]
French	[35]
Spanish	[35]
Bosnian	[35]
Croatian	[35]
Indonesian	[57]
Amharic	[11]

TABLE 5. Comparison of various models by rouge score.

Reference	Encoder⇔Decoder	Dataset	ROUGE Score
[21]	RNN⇔LSTM	DUC-2004, Gigaword Corpus	28.97
[25]	Bi-RNN⇔RNN	LCSTS	35.00
[31]	Transformer	Gigaword, Newsroom	38.72
[26]	Bi-LSTM⇔LSTM	CNN/Daily mail	39.53
[28]	Bi-LSTM⇔LSTM	CNN/Daily mail	41.22
[34]	Transformer	CNN/Daily mail, LCSTS	44.79

TABLE 6. Issues and challenges for abstractive summarization.

Issues and Challenges	Highlighted by	Addressed by
Scaling abstractive summarization systems	[20]	[23]
Efficient alignment	[20]	
Consistency in generation	[20]	
Rare word problem	[22]	
Out of vocabulary words		[23]
Factual details		[23]
Repetitive output		[26]
Efficient handling of images, audio, and video	[27]	
Slow speed		[29]
Inaccurate encoding of long docs		[29]
Long document summarization		[30]
Factual inconsistencies of abstractive summarization systems		[2]
Multilingual summarization		[35]
Multi-aspect (grammatical & semantic) evaluation metric	[53]	

D. DATASET AND EVALUATION METRIC

For the dataset and evaluation metric for abstractive summarization, it is obvious from Fig. 15 that many works in the survey utilize CNN/Daily mail dataset and ROUGE metric for summary evaluation. ROUGE works by comparing the n-gram similarity between the reference and the generated summary. Some researchers emphasize the need for a metric that can evaluate automatic summaries from grammatical aspects also.

Further examining the dataset element, we observe from this survey that some large datasets are available for the English language, some datasets for Chinese, German, and French also. But for other languages that are considered low resource languages, corpora are either not available or are composed of a very limited number of documents. Our discussion makes a connection with Table 4, which indicates languages across the survey for which abstractive summarization systems have been modeled.

It can be seen in Table 4 that most of the research, models English language summarization systems, some studies model Chinese language summarization systems while others took initiative on first creating datasets and then modeling systems for Amharic, Spanish, Indonesian, Bosnian, and Croatian languages.

E. MODEL COMPARISON BASED ON ROUGE SCORE

Table 5 provides a ROUGE score for six selected references to compare various models, by relating to their encoder-decoder architecture. Here, the highest ROUGE score of 44.79 points is achieved by [34], with the transformer-based encoder-decoder model. Reference [28] obtains a second-highest score of 41.22 ROUGE points by employing a bidirectional LSTM as encoder while a standard LSTM as the decoder.

From Table 5, we can compare [31] and [34] because both employed transformer-based architecture but the ROUGE score of [31] is considerably less than that of [34]. To understand the difference between ROUGE scores of [31] and [34], we compare other elements of the models from Fig. 15. First of all, we observe a major difference in the encoder column, as [31] employs a transformer encoder while in the case of [34], there exists a dual transformer encoder (BERT + standard transformer). Secondly, [31] implemented an attention mechanism while in the case of [34], the attention, as well as copying mechanism, is also utilized by the model. Also, both used ADAM optimizer. This comparison implies that the use of BERT and added functionality by introducing various mechanisms in addition to attention results in the improvement of the overall ROUGE score.

References [26] and [28] also provide an interesting comparison as both models utilize the same architecture, the same datasets, the same mechanisms but yield a variation in ROUGE scores. The only difference that can be observed from Fig. 15, is in the application of gradient clipping by [26]. However, looking deep into the paper [28], we find out that a different kind of attention known as bottom-up attention is incorporated in the model that probably resulted in an increased ROUGE score.

F. ISSUES AND CHALLENGES WITH ABSTRACTIVE SUMMARIZATION SYSTEMS

Despite several improvements in abstractive text summarization with neural networks over the recent years, these systems still pose some issues and challenges for the research community. To become familiar with current issues and providing solutions addressing these issues will result in more efficient and reliable summarization systems. The contents of Table 6 present some of the issues and challenges in neural networks-based abstractive text summarization which are highlighted and addressed across the reviewed literature.

VI. CONCLUSION

This article presents an up-to-date review of abstractive text summarization by surveying related scientific literature. A research framework is created in line with the key design elements like encoder-decoder architecture, mechanisms, training and optimization methods, datasets, and evaluation metric for the abstractive summarization models and is analyzed with the help of the concept matrix highlighting the common design trends of the recent abstractive summarization systems. Besides, the review provides an insight into various types of attention mechanisms, languages for which abstractive systems have been modeled, and issues and challenges associated with these systems, some of which have been addressed by researchers while others still need attention. Moreover, we have also highlighted some gaps like the use of pre-trained models like BART [10] and MASS [54] for abstractive summarization systems. The constraint of this study is the number of included papers. In future work, it can be updated by including more research on the topic. Also, a novel abstractive summarization system can be implemented utilizing the design elements identified in this study. As this study identifies various types of attention, so another idea for future work might be to conduct standalone experimental research on attention mechanisms for seq2seq models.

ACKNOWLEDGMENT

The authors would like to thank the reviewers of this article for their valuable suggestions and contribution in helping them to improve the quality of this work.

REFERENCES

[1] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D., J. B., and K. Kochut, "Text summarization techniques: A brief survey," *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 10, pp. 1–9, 2017, doi: 10.14569/ijacsa.2017.081052.

[2] Y. Zhang. (2019). *Evaluating the Factual Correctness for Abstractive Summarization*. [Online]. Available: https://www-cs.stanford.edu/~yuhuiz/assets/reports/factual.pdf

[3] A. Nenkova and K. McKeown, "Automatic summarization," in *Proc. 49th Annu. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol.*, Jun. 2011. [Online]. Available: https://repository.upenn.edu/cis_papers/710/

[4] E. Lloret and M. Palomar, "Text summarisation in progress: A literature review," *Artif. Intell. Rev.*, vol. 37, no. 1, pp. 1–41, Jan. 2012, doi: 10.1007/s10462-011-9216-z.

[5] H. Lin and V. Ng, "Abstractive summarization: A survey of the state of the art," in *Proc. 33rd AAAI Conf. Artif. Intell.*, vol. 33, pp. 9815–9822, 2019. [Online]. Available: https://www.aaai.org/ojs/index.php/AAAI/article/view/5056

[6] O. Klymenko, D. Braun, and F. Matthes, "Automatic text summarization: A state-of-the-art review," in *Proc. 22nd Int. Conf. Enterprise Inf. Syst.*, 2020, pp. 648–655.

[7] T. Shi, Y. Keneshloo, N. Ramakrishnan, and C. K. Reddy, "Neural abstractive text summarization with sequence-to-sequence models," 2018, *arXiv:1812.02303*. [Online]. Available: http://arxiv.org/abs/1812.02303

[8] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2019, pp. 4171–4186.

[9] A. Radford, "Improving language understanding by generative pre-training," *OpenAI J.*, to be published.

[10] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, doi: 10.18653/v1/2020.acl-main.703.

[11] A. M. Zaki, M. I. Khalil, and H. M. Abbas, "AMHARIC abstractive text summarization," *J. Xidian Univ.*, vol. 14, no. 6, pp. 1–5, 2020, doi: 10.37896/jxu14.6/094.

[12] H. P. Luhn, "The automatic creation of literature abstracts," *IBM J. Res. Develop.*, vol. 2, no. 2, pp. 159–165, Apr. 1958, doi: 10.1147/rd.22.0159.

[13] N. Rane and S. Govilkar, "Recent trends in deep learning based abstractive text summarization," *Int. J. Recent Technol. Eng.*, vol. 8, no. 3, Sep. 2019.

[14] Sciforce. (2019). *Towards Automatic Summarization. Part 2. Abstractive Methods*. Sciforce Blog. [Online]. Available: https://medium.com/sciforce/towards-automatic-summarization-part-2-abstractive-methods-c424386a65ea

[15] M. Sanad. (2019). *A Comprehensive Guide to Build your own Language Model in Python*. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-language-model-nlp-python-code/

[16] C. D. Manning, H. Schütze, and P. Raghavan, *Introduction to Information Retrieval*, vol. 238. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[17] K. Jing and J. Xu, "A survey on neural network language models," 2019, *arXiv:1906.03591*. [Online]. Available: http://arxiv.org/abs/1906.03591

[18] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Feb. 2003, doi: 10.1162/153244303322533223.

[19] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. 1st Int. Conf. Learn. Represent. (ICLR)*, 2013, pp. 1–12.

[20] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 379–389. [Online]. Available: https://arxiv.org/abs/1509.00685

[21] S. Chopra, M. Auli, and A. M. Rush, "Abstractive sentence summarization with attentive recurrent neural networks," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol.*, 2016, pp. 93–98.

[22] B. Hu, Q. Chen, and F. Zhu, "LCSTS: A large scale chinese short text summarization dataset," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 1967–1972, doi: 10.18653/v1/d15-1229.

[23] R. Nallapati, B. Zhou, C. dos Santos, C. Gulcehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence RNNs and beyond," in *Proc. 20th SIGNLL Conf. Comput. Natural Lang. Learn.*, 2016, pp. 280–290, doi: 10.18653/v1/k16-1028.

[24] Q. Chen, X. Zhu, Z. Ling, S. Wei, and H. Jiang, "Distraction-based neural networks for document summarization," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, 2016. [Online]. Available: https://arxiv.org/abs/1610.08462

[25] J. Gu, Z. Lu, H. Li, and V. O. K. Li, "Incorporating copying mechanism in sequence-to-sequence learning," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics (Long Papers)*, vol. 1, 2016, pp. 1631–1640, doi: 10.18653/v1/p16-1154.

[26] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics (Long Papers)*, vol. 1, 2017, pp. 1073–1083, doi: 10.18653/v1/P17-1099.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst., NIPS*, 2017, pp. 5998–6008. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[28] S. Gehrmann, Y. Deng, and A. Rush, "Bottom-up abstractive summarization," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 4098–4109, doi: 10.18653/v1/D18-1443.

[29] Y.-C. Chen and M. Bansal, "Fast abstractive summarization with reinforce-selected sentence rewriting," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics (Long Papers)*, vol. 1, 2018, pp. 675–686, doi: 10.18653/v1/p18-1063.

[30] D. Aksenov, "Abstractive text summarization with neural sequence-to-sequence models," M.S. thesis, 2019. [Online]. Available: https://www.politesi.polimi.it/bitstream/10589/153038/3/new_master_thesis_dmitrii_aksenov.pdf

[31] X. Duan, H. Yu, M. Yin, M. Zhang, W. Luo, and Y. Zhang, "Contrastive attention mechanism for abstractive sentence summarization," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, 2019, pp. 3044–3053, doi: 10.18653/v1/d19-1301.

[32] J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu, "PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization," *Proc. 37th Int. Conf. Mach. Learn.*, Vienna, Austria: PMLR, vol. 119, 2020, pp. 11328–11339.

[33] C. Zhu, R. Xu, M. Zeng, and X. Huang, "End-to-end abstractive summarization for meetings," in *Microsoft Speech and Dialogue Research Group*, 2020.

[34] I. Saito, K. Nishida, K. Nishida, A. Otsuka, H. Asano, J. Tomita, H. Shindo, and Y. Matsumoto, "Length-controllable abstractive summarization by guiding with summary prototype," 2020, *arXiv:2001.07331*. [Online]. Available: http://arxiv.org/abs/2001.07331

[35] Y. Cao, X. Wan, J. Yao, and D. Yu, "MultiSumm: Towards a unified model for multi-lingual abstractive summarization," in *Proc. 34th AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 11–18.

[36] A. K. M. Masum, S. Abujar, M. A. I. Talukder, A. K. M. S. A. Rabby, and S. A. Hossain, "Abstractive method of text summarization with sequence to sequence RNNs," in *Proc. 10th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2019, pp. 1–5.

[37] J. L. Elman, "Finding structure in time," *Cognit. Sci.*, vol. 14, no. 2, pp. 179–211, Mar. 1990, doi: 10.1016/0364-0213(90)90002-E.

[38] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1724–1734, doi: 10.3115/v1/d14-1179.

[39] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. 3rd Int. Conf. Learn. Represent., ICLR*, 2015, pp. 1–15.

[40] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[41] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 4, Sep. 2014, pp. 3104–3112. [Online]. Available: https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdfv

[42] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *Proc. NIPS Workshop Deep Learn.*, 2014.

[43] M. A. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," in *Proc. 4th Int. Conf. Learn. Represent.*, 2016, pp. 1–16.

[44] A. Venkatraman, M. Hebert, and J. A. Bagnell, *Improving Multi-Step Prediction of Learned Time Series Models*. Menlo Park, CA, USA: Association for the Advancement of Artificial Intelligence, 2015.

[45] S. Ruder, "An overview of gradient descent optimization algorithms," in *DBLP Computer Science Bibliography*, 2016.

[46] R. Khandelwal. (2019). *Overview of Different Optimizers for Neural Networks*. Retrieved From. [Online]. Available: https://medium.com/datadriveninvestor/overview-of-different-optimizers-for-neural-networks-e0ed119440c3

[47] M. Grusky, M. Naaman, and Y. Artzi, "Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol., (Long Papers)*, vol. 1, 2018, pp. 708–719.

[48] C. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Proc. Jpn. Circulat. J., Conf.*, vol. 34, 2004, p. 8.

[49] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics ACL*, 2001, pp. 311–318.

[50] S. Banerjee and A. Lavie, "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments," in *Proc. 2nd Workshop Stat. Machine Transl.*, 2007, pp. 65–72. [Online]. Available: https://www.aclweb.org/anthology/W05-0909.pdf

[51] J. Webster and R. T. Watson, "Analyzing the past to prepare for the future: Writing a literature review," *MIS Quart.*, vol. 26, no. 2, pp. 13–23, 2002.

[52] K. Al-Sabahi, Z. Zuping, and Y. Kang, "Bidirectional attentional encoder-decoder model and bidirectional beam search for abstractive summarization," 2018, *arXiv:1809.06662*. [Online]. Available: http://arxiv.org/abs/1809.06662

[53] N. Sanjabi, "Abstractive text summarization with attention-based mechanism," M.S. thesis, Artif. Intell., Bangalore, India, 2018. Available:. [Online]. Available: https://www.semanticscholar.org/paper/Abstractive-text-summarization-with-attention-based-Sanjabi/ee0153de49b6bed584327a64a5e960104b524b4e

[54] K. Song, X. Tan, T. Qin, J. Liu, and T. Liu, "MASS: Masked sequence to sequence pre-training for language generation," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 10384–10394.

[55] A. Esteva, A. Kale, R. Paulus, K. Hashimoto, W. Yin, D. Radev, and R. Socher, "CO-search: COVID-19 information retrieval with semantic search, question answering, and abstractive summarization," 2020, *arXiv:2006.09595*. [Online]. Available: http://arxiv.org/abs/2006.09595

[56] V. Kieuvongngam, B. Tan, and Y. Niu, "Automatic text summarization of COVID-19 medical research articles using BERT and GPT-2," 2020, *arXiv:2006.01997*. [Online]. Available: http://arxiv.org/abs/2006.01997

[57] R. Adelia, S. Suyanto, and U. N. Wisesty, "Indonesian abstractive text summarization using bidirectional gated recurrent unit," *Procedia Comput. Sci.*, vol. 157, pp. 581–588, Jan. 2019.

[58] Q. Wang and J. Ren, "Summary-aware attention for social media short text abstractive summarization," *Neurocomputing*, to be published, doi: 10.1016/j.neucom.2020.04.136.

[59] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–12. [Online]. Available: https://openreview.net/pdf?id=HkAClQgA-

[60] S. Li, D. Lei, P. Qin, and W. Y. Wang, "Deep reinforcement learning with distributional semantic rewards for abstractive summarization," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, 2019, pp. 6038–6044.

[61] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, 1997.

[62] S. Song, H. Huang, and T. Ruan, "Abstractive text summarization using LSTM-CNN based deep learning," *Multimedia Tools Appl.*, vol. 78, no. 1, pp. 857–875, Jan. 2019, doi: 10.1007/s11042-018-5749-3.

[63] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," *Sci. China Technol. Sci.*, vol. 63, no. 10, pp. 1872–1897, Oct. 2020, doi: 10.1007/s11431-020-1647-3.

AYESHA AYUB SYED received the bachelor's degree in computer engineering from the University of Engineering and Technology, Taxila, Pakistan, in 2008, and the master's degree in information systems management from Bina Nusantara University in 2018, where she is currently pursuing the Ph.D. degree with the Department of Doctor of Computer Science.

She was a Data Center Manager with the Special Communication Organization, Pakistan, from 2009 to 2010. Her research interests include embedded systems, information systems, e-commerce, deep learning, and automatic text summarization.

FORD LUMBAN GAOL (Senior Member, IEEE) received the B.Sc. degree in mathematics, the master's degree in computer science, and the Ph.D. degree in computer science from the University of Indonesia in 1997, 2001, and 2009, respectively. He is currently an Associate Professor in informatics engineering and information system with Bina Nusantara University, where he is also the Vice Chair of the Bina Nusantara University Doctorate Program with the Computer Science and Research Interest Group Leader for advance system in computational intelligence and knowledge engineering. He is the President of the IEEE Indonesia Section Computer Society and the Director of the IAIAI Southeast Asia Region. He was the Vice Chair of the IEEE Indonesia Section and the former Chair of the ACM Indonesia.

He was also an Invited Scholar with ICTP Trieste Italy in 2013 and Aligarh Muslim University in 2014. He is a Senior Member of the IEEE Computer Society, the IEEE Education Society, a member of the Indonesian Mathematical Society, a Professional Member of the Association for Computing Machinery Professional, the International Association of Engineers, and the Indonesia Society for Bioinformatics. He was a recipient of Visiting Professor with Kazan Federal University, Russia, in 2014 and 2015, Vladimir State University, Russia, in 2016, Financial University, Moscow, Russia, in 2017, Southern Federal University, Rostov-On Don, Russia, in 2018, the AIIT—Tokyo Metropolitan University in 2019. He has extensive networks on the scientific and professional societies, such as the IEEE TALE Steering Committee, the General Chair of the IEEE TALE 2013 and 2019, the Asian Conference on Intelligent Information and Database Systems Steering Committee, the General Chair of ACIIDS 2015 and 2019, the Advisory Council of Shridhar University, India, the IEEE Tencon 2011 Program Chair, the IEEE Tensymp 2016 General Chair, and the Co-Founder and the General Chair of the IEEE Comnetsat and the IEEE Cyberneticscom 2012 and 2013.

He is also a Ph.D. Supervisor for seven Ph.D. students in management science and two Ph.D. students in computer science and more than 50 master's students. He is supervising seven computer and information system Ph.D. students and 12 master's students. He is the Keynote Speaker of the Signal Processing and Information Communications in Bali Indonesia in 2020, the 5th International Conference on Intelligent Information Technology, Hanoi, Vietnam, in 2020, the 9th International Congress on Advanced Applied Informatics in Fukuoka, Japan, in 2020. He has 199 Scopus documents, with 208 citations and H index = 6.

TOKURO MATSUO (Member, IEEE) received the Ph.D. degree in engineering from the Department of Computer Science, Nagoya Institute of Technology, in 2006. He has been a Full Professor with the Advanced Institute of Industrial Technology since 2012.

He has been a Research Fellow with SEITI, Central Michigan University, USA, since 2010, the Executive Director with the International Institute of Applied Informatics since 2011, a Guest Professor with Bina Nusantara University, Indonesia, since 2015, a Research Project Professor with the Collective Intelligence Research Center, Nagoya Institute of Technology, Japan, since 2015, and an Invited Professor with the University of Nevada, Jas Vegas, USA, since 2016. He was a Visiting Researcher with the University of California at Irvine from 2010 to 2011, a research fellow with Shanghai University from 2010 to 2013, and a Research Project Professor with the Green Computing Research Center, Nagoya Institute of Technology, from 2011 to 2014.

His current research interests include electronic commerce and business, service science and marketing, business management, artificial intelligence, material informatics, tourism informatics, convention administration research, and incentive design on e-services.