2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE) 978-1-7281-4142-8/20/$31.00 ©2020 IEEE 10.1109/ic-ETITE47903.2020.087

Study on Abstractive Text Summarization Techniques

Parth Rajesh Dedhia, Hardik Pradeep Pachgade, Aditya Pradip Malani, Nataasha Raul, Meghana Naik
Sardar Patel Institute of Technology
Mumbai, India
parth.dedhia@spit.ac.in, hardik.pachgade@spit.ac.in, aditya.malani@spit.ac.in, nataasharaul@spit.ac.in, meghana_naik@spit.ac.in

***Abstract*—As there is an increase in the usage of digital applications, the availability of data generated has increased to a tremendous scale. Data is an important component in almost every domain where research and analysis are required to solve the problems. It is available in a structured or unstructured format. Therefore, in order to get corresponding data as per the application's purpose, easily and quickly from different sources of data on the internet, an online content summarizer is desired. Summarizers makes it easier for users to understand the content without reading it completely. Abstractive Text Summarizer helps in defining the content by considering the important words and helps in creating summaries that are in a human-readable format. The main aim is to make summaries in such a way that it should not lose its context. Various Neural Network models are employed along with other machine translation models to bring about a concise summary generation. This paper aims to highlight and study the existing contemporary models for abstractive text summarization and also to explore areas for further research.**

***Index Terms*—Abstractive Summarization, Attention, Seq2Seq, Encoder-Decoder, Pointer Mechanism**

I. Introduction

The amount of data available on the internet is increasing at an alarming rate. This data, on the internet, is unstructured. This unorganized form of data has made it difficult for a user to locate specific content. Eventually, the user spends time and resources to locate such relevant content. Hence, a content summarizer would be desirable to get a gist of an article and to check if the article answers the user's query before the user proceeds to read the article in detail. Automatic text Summarization is the basic and elementary methodology used for any text summarization systems [1], [2]. Automatic text summarizer has many applications in industry: News Aggregators, Blogs, Product Description, etc.

The Text Summarization Technique paved the way for the development of summarization models. Initially, the text summarization was developed considering basic parameters and measures. Text summarization can be classified as Abstractive and Extractive:

A. Extractive summarization

Extractive summarization finds out the important sections of the content and creates a subset of sentences from the sentences present in the original document. It does not add new words to the existing content and cannot combine two or more sentences to compact the content. Based on the number of documents used as sources, summarization is further classified as single-sourced and multi-sourced. Extractive Summarization basically works on the basis of combining the words or phrases from the corpus for the summary.

B. Abstractive summarization

Abstractive summarization, on the other hand, analyses the whole content to reproduce the original content in a new and optimized way using advanced natural language techniques. The newly generated content is shorter and more importantly, conveys the most critical information of the original content [3]. Abstractive summaries also generate fluent sentences that are grammatically correct, unlike extractive methods, which may lead to disfluent sentences.

This paper explains the basic workflow of abstractive text summarization along with the contemporary models. This paper is organized as follows: In section II, works related to extractive text summarization and their drawbacks are elaborated. The basics of Natural Language Processing(NLP) have been covered in section III. Different, existing models of abstractive text summarization are described in section IV. Table I provides a list of papers published on the topic of abstractive text summarization. Section V discusses further development along with the advantages of the proposed model in Section VI, and Section VII concludes the paper.

II. Related Work

Many domains and categories provide a user with considerable data but need summarization. One such category is news summarization explained in paper [4], which uses extractive text summarization for news articles. The algorithm creates a lexical chain from the given source article. A lexical chain is a logical group of words that depict the idea of the document. All the lexical chains are given a score by using a predefined scoring criterion. Then, individual sentences inside those chains are given a score whose score is greater than a minimum threshold. Again, those sentences are taken for generating a summary whose score is greater than a minimum threshold [4]. Alternatively, a feature extraction based approach can be used for extractive text summarization. A sentence in a document is processed based on seven predefined features, and each feature is given a value between 0 and 1.

A single rule is used to evaluate these seven features. All the sentences are then ordered, and the top sentences are used to generate a summary [5].

Apart from these, other approaches like a semantic graph-based approach, maximum marginal relevance approach, and the page rank method are also used for extractive text summarization. However, these summaries are extractive summaries, unlike those generated by humans, which leads to low readability. Furthermore, no new words are added in this process of creating summaries. Moreover, some level of abstraction was tried in extractive summaries. The graph-based method used for generating summaries was followed by a thesaurus to replace the words having a length greater than five letters. WordNet is used to get a set of sense of the word, and the WordRank algorithm picks up the word with the highest sense, as ranked by the algorithm. If the length of the obtained word is less than the one which is to be replaced, then the word is replaced [6].

Extractive text summarization has been used in a lot of applications, and a lot of research has been done on this topic, but few drawbacks have been highlighted [7]. Firstly, the summaries are not concise, as information is spread throughout the document. Hence, while extracting the information, some of it may be ignored. Moreover, in casual language, pronouns are highly used, and if these pronouns are not resolved, then the relevance of the pronoun is lost while summarizing the corpus. Sometimes, the corpus discusses conflicting ideas, and extractive text summaries can belie the user by providing the conflicting idea, and these ideas need to be resolved.

III. Abstractive Text Summarizer

To achieve abstractive text summarization, we first need to feed the computers with some text. Computers understand only binary numbers, and therefore it is necessary to feed some numeric data into the algorithm. This numeric data is nothing but the embedding of the words. The words in the documents are to be embedded irrespective of the summarization technique used. There are different ways of mapping the words in the text to real-valued vector representation, such as a one-hot encoding method for vectorizing the categorical features. One-Hot Encoding is a vector represented using binary numbers. Similarly, there are pre-trained embeddings such as word2vec and Glove. Word2Vec is a trained vector, which is a distribution of words in numerical features that are understood by the neural network. The words in the text are represented by semantic vector spaces models of the language by real-valued vectors [8]. Here the words are mapped into meaningful space based on the similarity of the words. Methods used in extractive text summarization are simple, but they fall short of making the text abstractive, as it may fail to grasp the important context or the essence of the document.

It is important to understand that text processing is done with the help of Recurrent Neural Network (RNN). It is a foundation laid for Text processing wherein the connection is formed between the nodes(word vectors) in sequential data

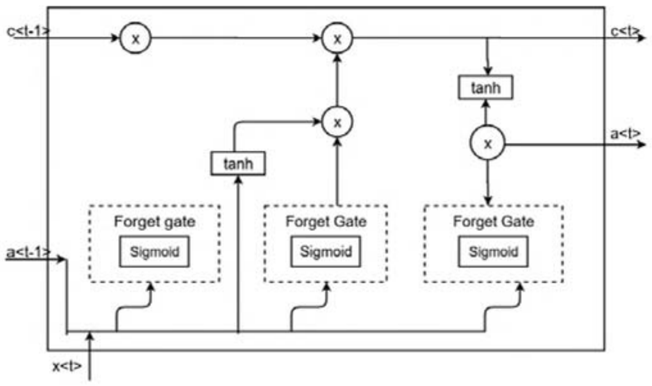

Fig. 1. LSTM Cell [10]

such as texts, speech, audio, etc. There are hidden layers in RNN, which helps in processing the earlier sequence of the data with the help of memory. This also helps the system to understand the context of the sequence. By leveraging RNN in the Sequence to Sequence (Seq2Seq) model, abstractive text summarization can be achieved [9]. Seq2Seq model considers the neighborhood of the current input word in the sequence and helps in generating the output sequence of words. There are varieties of RNN, and one of them is Long Short Term Memory(LSTM), which is explained further.

Given below is the algorithm used in [9]:

$$f_t = \sigma(W_{if}.E_{xt-1} + b_{if} + W_{hf}.h_{t-1} + b_{hf})(ForgetGate) \quad (1)$$

$$i_t = \sigma(W_{ii}.E_{xt-1} + b_{ii} + W_{hi}.h_{t-1} + b_{hi})(InputGate) \quad (2)$$

$$g_t = \tanh(W_{ig}.E_{xt-1} + b_{ig} + W_{hg}.h_{t-1} + b_{hg}) \quad (3)$$

$$c_t = f_t.c_{t-1} + i_t.g_t \quad (4)$$

$$o_t = \sigma(W_{io}.E_{xt-1} + b_{io} + W_{ho}h_{t-1} + b_{ho})(OutputGate) \quad (5)$$

$$h_t = o_t.\tanh(c_t) \quad (6)$$

E is the representation of the embeddings of the word from the input token x, and c is the representation of the cell states. Initially, h and c are initialized to 0. h is the hidden state of the cell at a particular time step t. Equation (1), the forget gate f_t decides what should be removed from the previous hidden state h_{t-1} and retaining what is required. W denotes the weight matrices, and b is the learning parameter vector. The sigmoid function helps in conceiving the value between the range of 0 and 1. Equation (2), the input gate decides to add new input values from the present or current input. The sigmoid functions help to determine which values to be updated from the present or previous input. From (3), the gate g_t helps in calculating the vector for the present (current) values and add it to the present cell state. c_t calculates the present cell state which is addition of $(input-gate*gate-gate)$ and $forget-gate$ given in (4). The output gate from 5 gives the output of the cell state by criticizing the values between -1 and 1, and then multiply it with an output of that sigmoid function explained in (6). These above equations are for determining cell state, candidate cell state along with the final output.

According to the bidirectional LSTM, the input encoded as $\vec{h}^e$ and h^e are represented as the encoded sequence of the input, here the forward and backward dependencies is denoted by arrows from left to right and right to left respectively. From this representation, *e* which is in the superscript of *h* indicates that the variable is used for encoder [9]. When the decoding takes place, it considers input as the encoded representation of the text data (i.e., hidden and cell states $\vec{h}_j^e$, h_l^e, $\vec{h}_j^e$, h_l^e) and then summary *y* is generated. The states used in LSTM decoders such as cell states and hidden states explained in (7) and (8) are as follows [9]:

$$h_0^d = \tanh(W_{e2d}(\vec{h}_j^e \oplus h_l^e) + b_{e2d}) \tag{7}$$

$$c_0^d = \vec{c}_j^e \oplus h^e \tag{8}$$

During every step of the decoding phase, the hidden state h_t^d is processed on the previous entities such as the cell and hidden states as shown in (9)

$$h_t^d = LSTM(h_{t-1}^d, E_{yt-1}) \tag{9}$$

In this way, the hidden states are passed to other parts of the model and accordingly, the probability weight-age of Vocabulary is calculated in (10) as follows-

$$P_{vocab}, t = softmax(W_{d2v} h_t^d + b_d 2v) \tag{10}$$

Here, the P_{vocab} is the probability of the target token which is represented as *w* being generated in the vocabulary represented as v. As per the survey by [9], the model's encoder is not trained well enough, because the similarity with the output is relatively different, due to which, the accuracy and readability of the generated summary are also very low.

LSTM was implemented for processing the sequential text data because, unlike primitive feedforward neural networks, it will not only help in singular node processing but also the complete sentence of the text. One of the implementations of this model is a vanilla Seq2Seq. In this model, the encoder and the decoder are implemented, where the encoder reads the input (11) from the source articles denoted as follows:

$$x = (x_1, x_2, ..., x_j), \tag{11}$$

and transforms it to hidden states

$$h^e = (h_1^e, h_2^e, ..., h_j^e) \tag{12}$$

while the decoder considers such hidden states (12) which will be the context inputs and helps in giving the output as the summary y,

$$y = (y_1, y_2, ..., y_T) \tag{13}$$

RNN architectures, especially the longest short term memory (LSTM) [10] and Gated recurrent unit (GRU) [11], are the most widely adopted approaches for seq2seq models [9]. These are essential because though RNN helps in catching the short dependencies, long-term context is not caught by simple RNN as it causes the diminishing gradient problem during backpropagation. This vanishing or diminishing gradient problem is caused as the training of the neural network becomes difficult due to gradient-based learning methods. One such method is backpropagation, where the network tries to reduce the error with the help of activation functions, but after certain iterations, the gradient becomes extremely small, and this affects the output of the network. The GRU as well as the LSTM [12] solves the issues of the gradients. Talking about GRU, it uses an updated and reset gates, which are the vectors helping in passing the information to the output. As a result, the information is retained for a long time.

Attention Mechanism has been a very important and successful mechanism used in seq2seq models for various tasks based on natural language processing [9] [13]. The attention architecture focuses on not only the state of the source article but also focuses on the parts of the content. This check happens on each step of the decoder. The mechanism of alignment is leveraged for achieving the attention [14].

In the encoder-decoder framework, the hidden states of the encoder are $h^e = (h_1^e, h_2^e, h_3^e, ..., h_j^e)$, the attention distribution calculated in (14) over the source token α_t^e is calculated as follows [9]:

$$\alpha_{tj}^e = \frac{exp(s_{tj}^e)}{\sum_{k=1}^{J} \exp(s_{tk}^e)} \tag{14}$$

where alignment score $s_{tj}^e = s(h_j^e, h_t^d)$ is obtained by the content based score function accordingly [15]:

$$s(h_j^e, h_t^d) = \left\{ \begin{array}{cc} (h_j^e)^T h_t^d, & dot \\ (h_j^e)^T_{align} h_t^d, & general \\ (v_{align})^T \tanh(W_{align}(h_j^e \oplus h_t^d) + b_{align}), & concat \end{array} \right\} \tag{15}$$

Here the dot represents the vector dimension. Apart from that, ‘general’ and ’concat’ functions are used mostly as the score functions in abstractive text summarization [16] [17]. The context vector of the source for the target word z as explained in (16):

$$z_t^e = \sum_{j=1}^{J} \alpha_{tj}^e h_j^e \tag{16}$$

Together with the current decoder hidden state h_t^d as given in (17), we get the attention hidden state [15] [9]:

$$h_t^d = W_z(z_t^e \oplus h_t^d) + b_z \tag{17}$$

Finally, the vocabulary distribution is calculated by [9]:

$$P_{vocab,t} = softmax(W_{d2v}h_t^d + b_{d2}) \tag{18}$$

When t>1, the decoder hidden state $h_{t+1}^d$is updated by:

$$h_{t+1}^d = LSTM(h_t^d, E_{yt} \oplus h_t^d) \tag{19}$$

Here,input passed in the LSTM function is the concatenation of E_{yt} and h_t^d [9]. A lot of development has been done on the attention model and is explained in the next section.

We take a little digression from the attention model and explain an altogether different concept of document context vector. When a person reads a document, an initial context is formed due to the title of the document, text in the web link, or search queries. Document Context Vectors (DCV) [18] try to model this human behavior in automatic text summarizers. DCV can be used to initialize the vectors of an RNN. The author of the content also enters some data about the article in the form of tags, image metadata, etc. These can also be used to create DCV's or to create a better impact, context vectors from all of the above sources can be combined by using appropriate weights. For a single source, say author, context vector (20)if built using-

$$C_s(w) = \left\{ \begin{array}{cc} frequency(w) in m_s, & if\ we\ seller\ metadata \\ 0, & otherwise \end{array} \right\} \tag{20}$$

where w is a word in the vocabulary
Next, we combine context vectors from all such sources to get- Then, Vector C_s is normalised to $\tilde{C_s}$.

$$C^d = \beta_s * \tilde{C_s^d} + \beta_q * \tilde{C_q^d} + \beta_b * \tilde{C_b^d} \tag{21}$$

Where β_s = 1, β_q = 1, β_b = 1. These parameters can be fine tuned as required. Then, value for each word dimension (w) in Cd is re-weighted by its IDF which is explained in (22)

$$C_i^d df(w) = C^d(w) * idf(w), for\ word\ w \tag{22}$$

Let M_{sgns} be a matrix of dimension Nxk where k is the dimension of word embedding and N is the vocabulary size. We get DCV Vd as given in (23),

$$v_d = C_i^d df * M_{SGNS} \tag{23}$$

Note that we get one V_d for every document, and its dimensions are 1xk, similar to word embeddings. In the next section, we move on to various developments done to the attention model as well as new methodologies introduce to make abstractive summarizes more readable.

IV. Model Architectures

As explained in the previous section, the Seq2Seq model is trained to read words of the input document using an encoder. The encoder can be a single-layered or may have more layers, thus making the network deeper and more abstract. The last encoded vector, which becomes the input to the decoder network, can be used by the decoder and can be used for generating summaries.

The model proposed in the paper [19] extends the state-of-the-art model for abstractive sentence summarization to recurrent neural network architecture. This model is a simplified version of the encoder-decoder framework for machine translation. In this case, the encoder input is a sequence of words which are subsequently converted into a vector representation and the decoder, assisted by the attention mechanism which focuses on specific words at each step of the input sequence [11]. The model is trained on the Gigaword corpus to generate headlines based on the first line of each news article. This model uses Conditional probability to generate a target sequence of tokens, which is calculated based on sentence-summary pairs. Convolutional and attention-based recurrent neural network model is used for the problem of abstractive sentence summarization [19].

The sequence to sequence approach, as discussed above, can be used for extractive text summarization [20], but can not be used directly for abstractive text summarization. It is observed that a variant of the sequence to sequence model, attention model [14], which was first designed and implemented for machine translation, can be successfully used for abstractive text summarization. The attention model implements a context vector that is trained to select important words and phrases. This selection happens by giving more weights to these phrases. The value of each of these weights is determined at the run time. Hence the model selects important words or phrases of the document under summarization.

$$s = \begin{bmatrix} \vec{h_1} \\ \vec{h_n} \end{bmatrix} \tag{24}$$

$$sGate_i = \sigma(W_s h_i + U_s s + b) \tag{25}$$

$$h_i' = h_i \cdot sGate_i \tag{26}$$

$$s_t = GRU(w_{t-1}, c_{t-1}, s_{t-1}) \tag{27}$$

$$s_0 = tanh(W_d \vec{h1} + b) \tag{28}$$

A novel approach was shown in [21], in which the encoder is made a layer deeper before implementing the attention model. The paper uses a *sGate*, where a dense layer is applied to the input of the encoded vector along with the sentence representation *s*. Equations (24), (25), and (26) show how the input vector passes through the sequence gate and h_i is generated for each input encoding. These new encodings generate the context vector c_t, and are used with the new hidden state s_t to find the predicted word for the next model. Equations (27) and (28) shows how the hidden state s_t is computed and then is used by the decoder to predict the summarized text. In the above equations W_s, U_s and W_d are trainable weight matrices.

$$r_t = W_r \vec{w_{t-1}} + U_r c_t + V_r s_t \tag{29}$$

$$m_t = [max(r_{t,2j-1}, r_{t,2j-1})] \tag{30}$$

$$p(y_t|y_1, y_2, ...y_{t-1}) = softmax(W_o m_t) \tag{31}$$

where W_r, U_r, V_r are the weight matrices. The output y_t generated here is used in predicting the next word.

Dual encoding [22], which is a modification of the attention model, is based on the human analogy that we read the same text again to comprehend it better. Similarly, the model reads the corpus two times, thus generating more concise summaries. The model uses a bidirectional GRU unit to encode the text during the first read and generates a context vector of this primary encoding. The context vector for the secondary encoding is generated from the following: Unidirectional GRU, hidden representation, and context vector of primary encoding. The decoder makes use of hidden vector encoding from primary encoder and context vector generated during primary encoding, and the secondary context vector which decoders partial sequence of fixed length K at each stage. Dual encoder makes use of pointer mechanism [23] as well as coverage mechanism [24] to obtain a readable summary.

In [25], explores a mechanism to remove redundancy in the generated summary by using the Convolutional Gated Unit, which performs global encoding on the source side information and reserves the meaning of core information. Also, it helps in filtering the words which may be repeated or does not have that much importance. Bidirectional LSTM encoder is implemented and Unidirectional LSTM decoder to read input words to generate word by word summary.

As suggested in [17], a bidirectional LSTM was used with a dual decoder. The decoder makes use of temporal attention over the simply encoded sequence, which penalizes the input token, which previously had high attention scores. The output tokens are generated by pointer mechanism [23] as well as by selecting the words out of vocabulary. This paper also suggested a novel metric of abstractive reward. This metric penalizes the model, forcing it to use out of vocabulary words and avoid copying from the source document. Thus, the metric avoids the summary from getting generated from the input document itself.

In the paper [26], the loss function for the intra-attention model is tweaked. The loss functions us multiplied by the difference between the baseline summary and the output summary, and positive loss is considered. So instead of maximizing the negative likely-hood, the authors minimized the new loss function. The authors introduced a hyper-parameter to a trade-off between the negative likelihood loss and the reinforced loss.

The first model for abstractive text summarization for single long Documents, wherein it considers Discourses for the hierarchical encoder. Discourses are the sections where the content size is large. For example, scientific papers are one such document having larger content as compared to other texts. The hierarchical encoder is nothing but an extension of RNN encoder. The encoder first encodes each discourse section, followed by encoding the whole document. The output of this encoder is stored as the final state of the entire sequence. The parameters for this encoder is shared for all discourse section. It is implemented with a single layer bidirectional LSTM. Afterward, Discourse Aware decoder [27] method is applied, wherein at each time step, it considers the discourse section, along with the words. For example, the abstract of any scientific or technical paper usually includes the description of the problem, discussion of the methods, final results, and conclusions. Here, all these sections are discourses, and hence discourse aware attentive model is used, wherein along with the words in the document the model, also looks for the section of the discourse heading. The decoder Coverage method is implemented, where they track attention coverage to avoid repeated usage of the same words.

$$p_{gen,t} = \sigma(W_{s,z} Z_t^e + W_{s,h} h_t^d + W_{s,E} Z_{y_{t-1}} + b_s) \tag{32}$$

The model described in paper [28] was modified to rich features during encoding by using parts of speech tags, named entity tags, Term Frequency(TF), and Inverse Document Frequency(IDF) tags. Due to this, a lookup based embedded matrix was created. In-order to recognize unseen, rare, or out of vocabulary words [29], a mechanism of pointing to a source document that contains that word, was implemented. Out of vocabulary words are handled by implementing a switch in the decoder network, as shown in (32), where, if the switch is off, the pointer points to a word in the source document. In the case when switch is off, the pointer points to attention distribution $\alpha_{t,j}^e$ by (14). To find the word from the input corpus following equation was used:

$$p_j = argmax_{j \in \{1,2,3,...,n\}} \alpha_{t,j}^e \tag{33}$$

where p_j is the position of the token in the source corpus. When the pointer is activated, the embedding which is present at location x_j, E_{x_j}. If the switch is on, then the word selected from (18). To make a summarization of huge text corpus, the model used hierarchical attention, where word-level attention and then a sentence level attention was used. The model was trained and tested on gig word, CNN daily mail, and DUC datasets. It was evaluated using ROUGE and the metrics used were full-length ROUGE F1 [30].

There are several proper nouns, phrases, and dates. These input during embedding, get represented as $\langle unk \rangle$ token. Since the model is trained in this way, the summaries produce $\langle unk \rangle$ tokens and hence, become unreadable. One approach to solve this problem was shown in paper [31]. The paper incorporates a copy mechanism to copy the words from the input text if an unknown token is seen as an output. A similar mechanism was also used in [32]. The pointer networks [23] were used to solve many other problems along with text summarization.

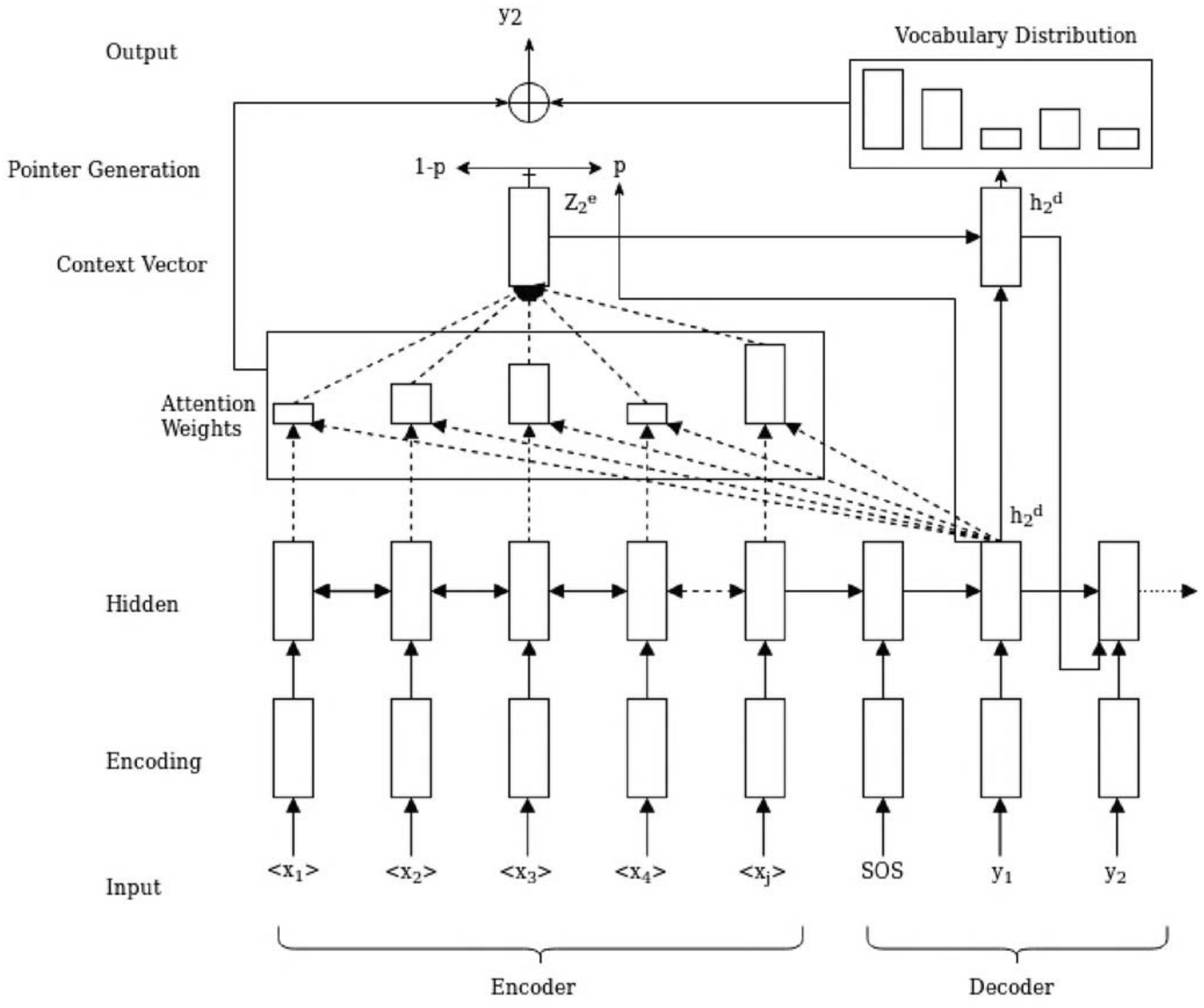

Fig. 2. Attention model with pointer mechanism

Fig. 2 shows the model of pointer mechanism along with the attention implemented in the paper [16]. Very similar to the mechanism implemented in paper [31], the vocabulary distribution of an extended vocabulary v_{ext} is calculated with the help of

$$P_{v_{ext}}(y_t) = p_{gen,t}P_g(y_t) + (1 - p_{gen,t})P_c(y_t) \tag{34}$$

where $p_{\text{gen,t}}$ is obtained from (32) and vocabulary distribution are defined as

$$P_g(y_t) = \left\{ \begin{array}{cc} P_{vocab,t}(y_t), & y_t \exists V_{ext} \cup <unk>. \\ 0, & Otherwise. \end{array} \right\} \tag{35}$$

and

$$P_c(y_t) = \left\{ \begin{array}{cc} \sum_{j:x_j=y_t} \alpha^e_{tj}, & y_t \exists X. \\ 0, & Otherwise. \end{array} \right\} \tag{36}$$

It is observed that p_{gen} is used as a soft switch if the values are between 0 and 1 and works as a hard switch if the value is either 0 or 1. It is used as a soft switch in [31] and used as a hard switch in the papers [32], [26], and [28].

V. Discussion

The text summarization has seen two approaches, extractive text summarization, and abstractive text summarization. Both abstractive and extractive text summarization can be used for various applications. However, people normally tend to read the summary that is more human-like i.e., abstractive rather than extractive. The proliferation of models of abstractive text summarization models using deep learning, clearly explains that a lot of research has already been done to produce more human-like summaries and the research is still pacing at an alarming rate with a sumptuous amount of papers being published each day.

TABLE I
COMPARISION OF VARIOUS MODELS

Year	*Referenced by*	*Methodology*	*Dataset*	*Metric*
2016	Chopra et al. [19]	Conditional recurrent neural network(RNN), novel convolutional attention-based encoder	DUC, GigaWord	Rouge
2016	Nallapati et al. [28]	Sequeuce to sequence attention model along with rare word using switching generator pointer was used.	DUC, Gigaword, CNN/DM	Rouge
2016	Gulcere et al. [32]	Two Softmax Layers	GigaWord	Rouge
2016	Gu et al. [31]	Copynet	LCSTS	Rouge
2017	See et al. [16]	Coverage Mechanism and Pointer Mechanism built on Seq2Seq	CNN/DM	Rouge/Meter
2017	Paulus et al. [26]	Neural intra Attentional Model on encoder decoder network,Hybrid Learning	CNN/DM, NYT	ROUGE,Human
2017	Zhou et al. [21]	Bidirectional GRU used for encoding, MLP for selection, GRU with selection criteria	DUC, GigaWord	ROUGE
2018	Lin et al. [25]	Global Encoding Framework using convolutional gated unit on attention based seq2seq model	LCSTS, GigaWord	ROUGE
2018	Cohan et al. [21]	Hierarchical encoder and discourse aware decoder to generate summary	PubMed,arXiv	ROUGE
2018	Kryściński et al. [17]	Decoder implemented with contextual Network and a pre-trained language model	CNN/DM	ROUGE and N-Gram metric
2018	Yao et al. [22]	Used 2 encoders and decoders to avoid word repetition	CNN,DUC	Rouge

The works for abstractive text summarization models got a boost after the attention model in [14], which was first implemented for machine translation. Almost all the papers since have tweaked the attention model. Most of the papers have made changes in the encoder, trying novel approaches, and producing better summaries. Similarly, many papers have also made changes to the way the attention mechanism is implemented for abstractive text summarization.

Although the summaries obtained from the above-tweaked models were condensed version of the corpus examined by the model, these models failed to produce human-readable summaries as many of the proper nouns, numbers, dates were not a part of the word embedding, thereby resulting with an unknown token. In-order to make summaries more readable, pointer mechanism was implemented and this was a mechanism to point out unknown words directly from the document and replace these unknown tokens with words from the input corpus. This made the summaries not only readable but also a concise version of the corpus.

There has been sparse work on deciding how the model of RNN will be initialized for its hidden input for the first RNN unit. Document Context Vector as explained in [18] could be used as an approach to initialize the vector input for the hidden unit of the first RNN.

VI. Advantages

The model suggested by us shows significant improvement over the currently existing models. The use of Document Context Vectors (DCV) to initialize the RNN instead of keeping the initialization vector empty allows our model to take into account the context of the metadata of the article like heading, URLs, meta-tags, etc. while generating the summary. Our proposed model could make use of the pointer mechanism to overcome the problem of rare words faced by other contemporary models. Pointers could be used to pay special attention to the rare words while generating the summary.

VII. Conclusion and Future Scope

This paper has explained both extractive and abstractive and taken a deeper look at abstractive text summarization. Abstractive Text Summarization has been one of the most successful applications of the Sequence to Sequence model and has gained a lot of attention in industry as well as academia. This paper has explained the basis of RNN models used for the development of attention models, as well as gives a brief insight into feature selection, attention model, pointer mechanism, and how in synergy they produce abstractive text summaries. The current model does not work if multiple documents are passed to the model. In the future, a mechanism can be devised to preserve the context of the previous document before taking into account the next document.

References

[1] O. Sornil and K. Gree-ut, "An automatic text summarization approach using content-based and graph-based characteristics," in *2006 IEEE Conference on Cybernetics and Intelligent Systems*, June 2006, pp. 1–6.

[2] P. Gupta, R. Tiwari, and N. Robert, "Sentiment analysis and text summarization of online reviews: A survey," in *2016 International Conference on Communication and Signal Processing (ICCSP)*, April 2016, pp. 0241–0245.

[3] M. N. Uddin and S. A. Khan, "A study on text summarization techniques and implement few of them for bangla language," in *2007 10th international conference on computer and information technology*, Dec 2007, pp. 1–4.

[4] P. Sethi, S. Sonawane, S. Khanwalker, and R. B. Keskar, "Automatic text summarization of news articles," in *2017 International Conference on Big Data, IoT and Data Science (BID)*, Dec 2017, pp. 23–29.

[5] S. S. Naik and M. N. Gaonkar, "Extractive text summarization by feature-based sentence extraction using rule-based concept," in *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, May 2017, pp. 1364–1368.

[6] A. P. Patil, S. Dalmia, S. A. A. Ansari, T. Aul, and V. Bhatnagar, "Automatic text summarizer," in *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sep. 2014, pp. 1530–1534.

[7] N. Kasture, N. Yargal, N. N. Singh, N. Kulkarni, and V. Mathur, "A survey on methods of abstractive text summarization," *Int. J. Res. Merg. Sci. Technol*, vol. 1, no. 6, pp. 53–57, 2014.

[8] R. S. Jeffrey Pennington and C. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. [Online]. Available: https://www.aclweb.org/anthology/D14-1162

[9] T. Shi, Y. Keneshloo, N. Ramakrishnan, and C. K. Reddy, “Neural abstractive text summarization with sequence-to-sequence models,” *arXiv preprint arXiv:1812.02303*, 2018.

[10] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[11] 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,” *arXiv e-prints*, p. arXiv:1406.1078, Jun 2014.

[12] N. Raphal, H. Duwarah, and P. Daniel, “Survey on abstractive text summarization,” in *2018 International Conference on Communication and Signal Processing (ICCSP)*, April 2018, pp. 0513–0517.

[13] D. Hu, “An introductory survey on attention mechanisms in nlp problems,” in *Intelligent Systems and Applications*, Y. Bi, R. Bhatia, and S. Kapoor, Eds. Cham: Springer International Publishing, 2020, pp. 432–448.

[14] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.

[15] T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 1412–1421. [Online]. Available: https://www.aclweb.org/anthology/D15-1166

[16] A. See, P. J. Liu, and C. D. Manning, “Get to the point: Summarization with pointer-generator networks,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 1073–1083. [Online]. Available: https://www.aclweb.org/anthology/P17-1099

[17] W. Kryściński, R. Paulus, C. Xiong, and R. Socher, “Improving abstraction in text summarization,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 1808–1817. [Online]. Available: https://www.aclweb.org/anthology/D18-1207

[18] C. Khatri, G. Singh, and N. Parikh, “Abstractive and Extractive Text Summarization using Document Context Vector and Recurrent Neural Networks,” *arXiv e-prints*, p. arXiv:1807.08000, Jul 2018.

[19] S. Chopra, M. Auli, and A. M. Rush, “Abstractive sentence summarization with attentive recurrent neural networks,” in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 93–98. [Online]. Available: https://www.aclweb.org/anthology/N16-1012

[20] R. Nallapati, F. Zhai, and B. Zhou, “Summarunner: A recurrent neural network based sequence model for extractive summarization of documents,” in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, ser. AAAI'17. AAAI Press, 2017, p. 3075–3081.

[21] Q. Zhou, N. Yang, F. Wei, and M. Zhou, “Selective encoding for abstractive sentence summarization,” *arXiv preprint arXiv:1704.07073*, 2017.

[22] K. Yao, L. Zhang, D. Du, T. Luo, L. Tao, and Y. Wu, “Dual encoding for abstractive text summarization,” *IEEE Transactions on Cybernetics*, vol. 50, no. 3, pp. 985–996, March 2020.

[23] O. Vinyals, M. Fortunato, and N. Jaitly, “Pointer networks,” in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 2692–2700. [Online]. Available: http://papers.nips.cc/paper/5866-pointer-networks.pdf

[24] Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li, “Modeling coverage for neural machine translation,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 76–85. [Online]. Available: https://www.aclweb.org/anthology/P16-1008

[25] J. Lin, X. Sun, S. Ma, and Q. Su, “Global encoding for abstractive summarization,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 163–169. [Online]. Available: https://www.aclweb.org/anthology/P18-2027

[26] R. Paulus, C. Xiong, and R. Socher, “A deep reinforced model for abstractive summarization,” in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=HkAClQgA-

[27] A. Cohan, F. Dernoncourt, D. S. Kim, T. Bui, S. Kim, W. Chang, and N. Goharian, “A discourse-aware attention model for abstractive summarization of long documents,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 615–621. [Online]. Available: https://www.aclweb.org/anthology/N18-2097

[28] R. Nallapati, B. Zhou, C. dos Santos, Ç. GuÌ‡lçehre, and B. Xiang, “Abstractive text summarization using sequence-to-sequence RNNs and beyond,” in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 280–290. [Online]. Available: https://www.aclweb.org/anthology/K16-1028

[29] S. Jean, K. Cho, R. Memisevic, and Y. Bengio, “On using very large target vocabulary for neural machine translation,” in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 1–10. [Online]. Available: https://www.aclweb.org/anthology/P15-1001

[30] E. ShafieiBavani, M. Ebrahimi, R. Wong, and F. Chen, “A semantically motivated approach to compute rouge scores,” *arXiv preprint arXiv:1710.07441*, 2017.

[31] J. Gu, Z. Lu, H. Li, and V. O. Li, “Incorporating copying mechanism in sequence-to-sequence learning,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1631–1640. [Online]. Available: https://www.aclweb.org/anthology/P16-1154

[32] C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio, “Pointing the unknown words,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 140–149. [Online]. Available: https://www.aclweb.org/anthology/P16-1014