Neural Abstractive Text Summarization with Sequence-to-Sequence Models

TIAN SHI, YASER KENESHLOO, NAREN RAMAKRISHNAN, and CHANDAN K. REDDY,
Virginia Tech

In the past few years, neural abstractive text summarization with sequence-to-sequence (seq2seq) models have gained a lot of popularity. Many interesting techniques have been proposed to improve seq2seq models, making them capable of handling different challenges, such as saliency, fluency and human readability, and generate high-quality summaries. Generally speaking, most of these techniques differ in one of these three categories: network structure, parameter inference, and decoding/generation. There are also other concerns, such as efficiency and parallelism for training a model. In this article, we provide a comprehensive literature survey on different seq2seq models for abstractive text summarization from the viewpoint of network structures, training strategies, and summary generation algorithms. Several models were first proposed for language modeling and generation tasks, such as machine translation, and later applied to abstractive text summarization. Hence, we also provide a brief review of these models. As part of this survey, we also develop an open source library, namely, Neural Abstractive Text Summarizer (NATS) toolkit, for the abstractive text summarization. An extensive set of experiments have been conducted on the widely used CNN/Daily Mail dataset to examine the effectiveness of several different neural network components. Finally, we benchmark two models implemented in NATS on the two recently released datasets, namely, Newsroom and Bytecup.

CCS Concepts: • **Information systems** → **Summarization**; • **Computing methodologies** → **Natural language processing**; **Natural language generation**; **Neural networks**; • **Theory of computation** → *Reinforcement learning*;

Additional Key Words and Phrases: Abstractive text summarization, sequence-to-sequence models, attention model, pointer-generator network, deep reinforcement learning, beam search

ACM Reference format:
Tian Shi, Yaser Keneshloo, Naren Ramakrishnan, and Chandan K. Reddy. 2020. Neural Abstractive Text Summarization with Sequence-to-Sequence Models. *ACM/IMS Trans. Data Sci.* 2, 1, Article 1 (December 2020), 37 pages.
https://doi.org/10.1145/3419106

1 INTRODUCTION

In the modern era of big data, retrieving useful information from a large number of textual documents is a challenging task, due to the unprecedented growth in the availability of blogs, news articles, and reports is explosive. Automatic text summarization provides an effective solution

This work was supported in part by the US National Science Foundation grants IIS-1619028, IIS-1707498, and IIS-1838730.
Authors' address: T. Shi, Y. Keneshloo, N. Ramakrishnan, and C. K. Reddy, Virginia Tech, 900 N Glebe Road, Arlington, VA 22203; emails: {tshi, yaserkl}@vt.edu, {naren, reddy}@cs.vt.edu.

2577-3224/2020/12-ART1 $15.00
https://doi.org/10.1145/3419106

for summarizing these documents. The task of the text summarization is to condense long documents into short summaries while preserving the important information and meaning of the documents [1, 111]. Having such short summaries, the text content can be retrieved, processed and digested effectively and efficiently. Generally speaking, there are two ways to perform text summarization: Extractive and Abstractive [90]. A method is considered to be *extractive* if words, phrases, and sentences in the summaries are selected from the source articles [1, 12, 30, 38, 87, 102, 117, 135]. They are relatively simple and can produce grammatically correct sentences. The generated summaries usually persist salient information of source articles and have a good performance w.r.t. human-written summaries [119, 135, 144, 158]. On the other hand, abstractive text summarization has attracted much attention since it is capable of generating novel words using language generation models conditioned on representation of source documents [98, 116]. Thus, they have a strong potential of producing high-quality summaries that are verbally innovative and can also easily incorporate external knowledge. In this category, many deep neural network based models have achieved better performance in terms of the commonly used evaluation measures (such as *ROUGE* [80] score) compared to traditional extractive approaches [15, 109]. In this article, we primarily focus on the recent advances of recurrent-neural-network (RNN)-based sequence-to-sequence (seq2seq) models for the task of abstractive text summarization.

1.1 RNN-Based Seq2seq Models and Pointer-Generator Network

Seq2seq models (see Figure 2) [21, 127] have been successfully applied to a variety of natural language processing (NLP) tasks, such as machine translation [3, 66, 89, 122, 145], headline generation [22, 116, 123], text summarization [98, 119], and speech recognition [4, 43, 94]. Inspired by the success of neural machine translation (NMT) [3], Rush et al. [116] first introduced a neural attention seq2seq model with an attention-based encoder and a Neural Network Language Model (NNLM) decoder to the abstractive sentence summarization task, which has achieved a significant performance improvement over conventional methods. Chopra et al. [22] further extended this model by replacing the feed-forward NNLM with a recurrent neural network (RNN). The model is also equipped with a convolutional attention-based encoder and a RNN (Elman [35] or LSTM [52]) decoder, and outperforms other state-of-the-art models on commonly used benchmark datasets, i.e., the Gigaword corpus. Nallapati et al. [98] introduced several novel elements to the RNN encoder-decoder architecture to address critical problems in the abstractive text summarization, including using the following: (i) feature-rich encoder to capture keywords, (ii) a switching generator-pointer to model out-of-vocabulary (OOV) words, and (iii) the hierarchical attention to capture hierarchical document structures. They also established benchmarks for these models on a CNN/Daily Mail dataset [16, 50], which consists of pairs of news articles and multi-sentence highlights (summaries). Before this dataset was introduced, many abstractive text summarization models have concentrated on compressing short documents to single sentence summaries [22, 116]. For the task of summarizing long documents into multi-sentence summaries, these models have several shortcomings: (1) They cannot accurately reproduce the salient information of source documents. (2) They cannot efficiently handle OOV words. (3) They tend to suffer from word- and sentence-level repetitions and generating unnatural summaries. To tackle the first two challenges, See et al. [119] proposed a pointer-generator network that implicitly combines the abstraction with the extraction. This pointer-generator architecture can copy words from source texts via a pointer and generate novel words from a vocabulary via a generator. With the pointing/copying mechanism [20, 45, 46, 93, 125, 137, 152], factual information can be reproduced accurately and OOV words can also be taken care in the summaries. Many subsequent studies that achieved state-of-the-art performance have also demonstrated the effectiveness of the pointing/copying mechanism [15, 41, 59, 109]. The third problem has been addressed by the coverage mechanism [119], intra-temporal and intra-decoder attention mechanisms [109], and some other heuristic approaches, like forcing a decoder to never output the same trigram more than once during testing [109].

1.2 Training Strategies

There are two other non-trivial issues with the current seq2seq framework, i.e., *exposure bias* and *inconsistency of training and testing measurements* [8, 62, 113, 134]. Based on the neural probabilistic language model [9], seq2seq models are usually trained by maximizing the likelihood of ground-truth tokens given their previous ground-truth tokens and hidden states (Teacher Forcing algorithm [8, 143], see Figure 4(b)). However, at testing time (see Figure 4(a)), previous ground-truth tokens are unknown, and they are replaced with tokens generated by the model itself. Since the generated tokens have never been exposed to the decoder during training, the decoding error can accumulate quickly during the sequence generation. This is known as *exposure bias* [113]. The other issue is the *mismatch of measurements*. Performance of seq2seq models is usually estimated with non-differentiable evaluation metrics, such as ROUGE [80] and BLEU [105] scores, which are inconsistent with the log-likelihood function (cross-entropy loss) used in the training phase. These problems are alleviated by the curriculum learning and reinforcement learning (RL) approaches.

1.2.1 Training with Curriculum and Reinforcement Learning Approaches. Bengio et al. [8] proposed a curriculum learning approach, known as *scheduled sampling*, to slowly change the input of the decoder from ground-truth tokens to model generated ones. Thus, the proposed meta-algorithm bridges the gap between training and testing. It is a practical solution for avoiding the exposure bias. Ranzato et al. [113] proposed a sequence level training algorithm, called MIXER (Mixed Incremental Cross-Entropy Reinforce), which consists of the cross entropy training, REINFORCE [142] and curriculum learning [8]. REINFORCE can make use of any user-defined task specific reward (e.g., non-differentiable evaluation metrics), therefore, combining with curriculum learning, the proposed model is capable of addressing both issues of seq2seq models. However, REINFORCE suffers from the high variance of gradient estimators and instability during training [2, 114, 145]. Bahdanau et al. [2] proposed an actor-critic-based RL method which has relatively lower variance for gradient estimators. In the actor-critic method, an additional critic network is trained to compute value functions given the policy from the actor network (a seq2seq model), and the actor network is trained based on the estimated value functions (assumed to be exact) from the critic network. On the other hand, Rennie et al. [114] introduced a self-critical sequence training method (SCST) which has a lower variance compared to REINFORCE and does not need the second critic network.

1.2.2 Applications to Abstractive Text Summarization. RL algorithms for training seq2seq models have achieved success in a variety of language generation tasks, such as image captioning [114], machine translation [2], and dialogue generation [77]. Specific to the abstractive text summarization, Ling and Rush [82] introduced a coarse-to-fine attention framework for the purpose of summarizing long documents. Their model parameters were learned with REINFORCE algorithm. Zhang and Lapata [155] used REINFORCE algorithm and the curriculum learning strategy for the sentence simplification task. Paulus et al. [109] first applied the self-critic policy gradient algorithm to training their seq2seq model with the copying mechanism and obtained the state-of-the-art performance in terms of ROUGE scores [80]. They proposed a mixed objective function that combines the RL loss with the traditional cross-entropy loss. Thus, their method can both leverage the non-differentiable evaluation metrics and improve the readability. Celikyilmaz et al. [15] introduced a novel deep communicating agents method for abstractive summarization, where they also adopted the RL loss in their objective function. Pasunuru et al. [107] applied the self-critic policy gradient algorithm to train the pointer-generator network. They also introduced two novel rewards (i.e., saliency and entailment rewards) in addition to ROUGE metric to keep the generated summaries salient and logically entailed. Li et al. [78] proposed a training framework based on the actor-critic method, where the actor network is an attention-based seq2seq model, and the critic network consists of a maximum likelihood estimator and a global summary quality estimator that is used to distinguish the generated and ground-truth summaries via a neural network binary classifier. Chen et al. [19] proposed a compression-paraphrase multi-step procedure, for abstractive text summarization, which first extracts salient sentences from documents and then rewrites them. In their model, they used an advantage actor-critic algorithm to optimize the sentence extractor for a better extraction strategy. Keneshloo et al. [62] conducted a comprehensive summary of various RL methods and their applications in training seq2seq models for different NLP tasks. They also implemented these RL algorithms in an open source library (https://github.com/yaserkl/RLSeq2Seq/) constructed using the pointer-generator network [119] as the base model.

1.3 Beyond RNN-Based Seq2Seq Models

Most of the prevalent seq2seq models that have attained state-of-the-art performance for sequence modeling and language generation tasks are RNN, especially long short-term memory (LSTM) [52] and gated recurrent unit (GRU) [23], based encoder-decoder models [3, 127]. Standard RNN models are difficult to train due to the vanishing and exploding gradients problems [10]. LSTM is a solution for vanishing gradients problem, but still does not address the exploding gradients issue. This issue is recently solved using a gradient norm clipping strategy [106]. Another critical problem of RNN-based models is the computation constraint for long sequences due to their inherent sequential dependence nature. In other words, the current hidden state in an RNN is a function of previous hidden states. Because of such dependence, RNN cannot be parallelized within a sequence along the time-step dimension (see Figure 2) during training and evaluation, and hence training them becomes major challenge for long sequences due to the computation time and memory constraints of GPUs [133].

Recently, it has been found that the convolutional neural network (CNN)-based [69] encoder-decoder models have the potential to alleviate the aforementioned problem, since they have better performance in terms of the following three considerations [14, 40, 60]: (1) A model can be parallelized during training and evaluation; (2) The computational complexity of the model is linear with respect to the length of sequences; and (3) The model has short paths between pairs of input and output tokens, so that it can propagate gradient signals more efficiently [51]. Kalchbrenner et al. [60] introduced a *ByteNet* model which adopts the one-dimensional convolutional neural network of fixed depth to both the encoder and the decoder [132]. The decoder CNN is stacked on top of the hidden representation of the encoder CNN, which ensures a shorter path between input and output. The proposed ByteNet model has achieved state-of-the-art performance on a character-level machine translation task with parallelism and linear-time computational complexity [60]. Bradbury et al. [14] proposed a quasi-recurrent neural network (QRNN) encoder-decoder architecture, where both encoder and decoder are composed of convolutional layers and so-called “dynamic average pooling” layers [7, 14]. The convolutional layers allow computations to be completely parallel across both mini-batches and sequence time-step dimensions, while they require less amount of time compared with computation demands for LSTM, despite the sequential dependence still presents in the pooling layers [14]. This framework has demonstrated to be effective by outperforming LSTM-based models on a character-level machine translation task with a significantly higher computational speed. Recently, Gehring et al. [39, 40] attempted to build CNN-based seq2seq models and apply them to large-scale benchmark datasets for sequence modeling. In [39], the authors proposed a convolutional encoder model, in which the encoder is composed of a succession of convolutional layers, and demonstrated its strong performance for machine translation. They further constructed a convolutional seq2seq architecture by replacing the LSTM decoder with a CNN decoder and bringing in several novel elements, including gated linear units [31] and multi-step attention [40]. The model also enables computations of all network elements parallelized, thus training and decoding can be much faster than the RNN models. It also achieved competitive performance on several machine translation benchmark datasets. More rencently, ConvS2S model [37, 40] has been applied to the abstractive document summarization and outperforms the pointer-generator network [119] on the CNN/Daily Mail dataset.

Vaswani et al. [133] constructed a novel network architecture called Transformer which only depends on feed-forward networks and a multi-head attention mechanism. It has achieved superior performance in machine translation task with significantly less training time. Currently, large Transformers [32, 133, 149], which are pre-trained on a massive text corpus with self-supervised objectives, have achieved superior results in a variety of downstream NLP tasks such as machine understanding [32, 84], question-answering [27, 86], and abstractive text summarization [34, 72, 85, 112, 148, 153]. Zhang et al. [153] demonstrated that their pre-trained encoder-decoder model can outperform previous state-of-the-art results [28, 36, 44, 63, 67, 99, 100, 119, 121] on several datasets by fine-tuning with limited supervised examples, which shows that pre-trained models are promising candidates in zero-shot and low-resource summarization tasks.

1.4 Other Studies

So far, we primarily focused on the pointer-generator network, training neural networks with RL algorithms, CNN based seq2seq architectures, and Transformers. There are many other studies that aim to improve the performance of RNN seq2seq models for the task of abstractive text summarization from different perspectives and broaden their applications.

1.4.1 Network Structure and Attention. The first way to boost the performance of seq2seq models is to design better network structures. Zhou et al. [159] introduced an information filter, namely, a selective gate network between the encoder and decoder. This model can control the information flow from the encoder to the decoder via constructing a second level representation of the source texts with the gate network. Zeng et al. [152] introduced a read-again mechanism to improve the quality of the representations of the source texts. Tan et al. [130] built a graph-ranking model upon a hierarchical encoder-decoder framework, which enables the model to capture the salient information of the source documents and generate accurate, fluent, and non-redundant summaries. Xia et al. [146] proposed a deliberation network that passes the decoding process multiple times (deliberation process), to polish the sequences generated by the previous decoding process. Li et al. [79] incorporated a sequence of variational auto-encoders [65, 115] into the decoder to capture the latent structure of the generated summaries.

1.4.2 Extraction + Abstraction. Another way to improve the abstractive text summarization is to make use of the salient information from the extraction process. Hsu et al. [54] proposed a unified framework that takes advantage of both extractive and abstractive summarization using a novel attention mechanism, which is a combination of the sentence-level attention (based on the extractive summarization [97]) and the word-level attention (based on the pointer-generator network [119]), inspired by the intuition that words in less attended sentences should have lower attention scores. Chen and Bansal [19] introduced a multi-step procedure, namely compression-paraphrase, for abstractive summarization, which first extracts salient sentences from documents and then rewrites them in order to get final summaries. Li et al. [73] introduced a guiding generation model, where the keywords in source texts is first retrieved with an extractive model [95].

Then, a guide network is applied to encode them to obtain the key information representations that will guide the summary generation process.

1.4.3 Long Documents. Compared to short articles and texts with moderate lengths, there are many challenges that arise in long documents, such as difficulty in capturing the salient information [28]. Nallapati et al. [98] proposed a hierarchical attention model to capture hierarchical structures of long documents. To make models scale-up to very long sequences, Ling and Rush [82] introduced a coarse-to-fine attention mechanism, which hierarchically reads and attends long documents (A document is split into many chunks of texts.). By stochastically selecting chunks of texts during training, this approach can scale linearly with the number of chunks instead of the number of tokens. Cohan et al. [28] proposed a discourse-aware attention model which has a similar idea to that of a hierarchical attention model. Their model was applied to two large-scale datasets of scientific papers, i.e., arXiv and PubMed datasets. Tan et al. [130] introduced a graph-based attention model which is built upon a hierarchical encoder-decoder framework where the pagerank algorithm [104] was used to calculate saliency scores of sentences.

1.4.4 Multi-Task Learning. Multi-task learning has become a promising research direction for this problem since it allows seq2seq models to handle different tasks. Pasunuru et al. [108] introduced a multi-task learning framework, which incorporates knowledge from an entailment generation task into the abstractive text summarization task by sharing decoder parameters. They further proposed a novel framework [47] that is composed of two auxiliary tasks, i.e., question generation and entailment generation, to improve their model for capturing the saliency and entailment for the abstractive text summarization. In their model, different tasks share several encoder, decoder and attention layers. McCann et al. [92] introduced a Natural Language Decathlon, a challenge that spans ten different tasks, including question-answering, machine translation, summarization, and so on. They also proposed a multitask question answering network that can jointly learn all tasks without task-specific modules or parameters, since all tasks are mapped to the same framework of question-answering over a given context.

1.4.5 Beam Search. Beam search algorithms have been commonly used in the decoding of different language generation tasks [119, 145]. However, the generated candidate-sequences are usually lacking in diversity [42]. In other words, top-K candidates are nearly identical, where K is size of a beam. Li et al. [74] replaced the log-likelihood objective function in the neural probabilistic language model [9] with Maximum Mutual Information (MMI) [5] in their neural conversation models to remedy the problem. This idea has also been applied to neural machine translation (NMT) [75] to model the bi-directional dependency of source and target texts. They further proposed a simple yet fast decoding algorithm that can generate diverse candidates and has shown performance improvement on the abstractive text summarization task [76]. Vijayakumar et al. [136] proposed generating diverse outputs by optimizing for a diversity-augmented objective function. Their method, referred to as the *Diverse Beam Search (DBS) algorithm*, has been applied to image captioning, machine translation, and visual question-generation tasks. Cibils et al. [26] introduced a meta-algorithm that first uses DBS to generate summaries, and then, picks candidates according to maximal marginal relevance [48] under the assumption that the most useful candidates should be close to the source document and far away from each other. The proposed algorithm has boosted the performance of the pointer-generator network on CNN/Daily Mail dataset.

Despite many research papers that are published in the area of neural abstractive text summarization, there are few survey papers [29, 96, 110] that provide a comprehensive study. In this article, we systematically review current advances of seq2seq models for the abstractive text summarization task from various perspectives, including network structures, training strategies, and

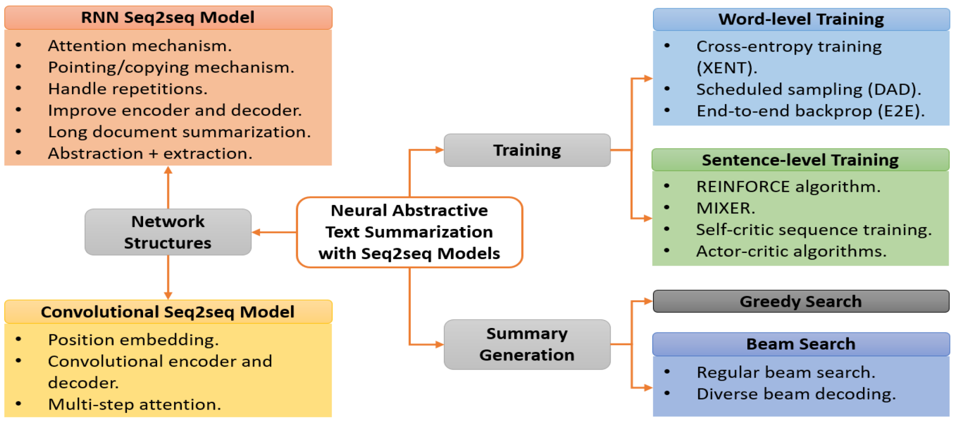
RNN Seq2seq Model
Attention mechanism.
Pointing/copying mechanism.
Handle repetitions.
Improve encoder and decoder.
Long document summarization.
Abstraction + extraction.
Network Structures
Convolutional Seq2seq Model
Position embedding.
Convolutional encoder and decoder.
Multi-step attention.
Neural Abstractive Text Summarization with Seq2seq Models
Training
Summary Generation
Word-level Training
Cross-entropy training (XENT).
Scheduled sampling (DAD).
End-to-end backprop (E2E).
Sentence-level Training
REINFORCE algorithm.
MIXER.
Self-critic sequence training.
Actor-critic algorithms.
Greedy Search
Beam Search
Regular beam search.
Diverse beam decoding.

Fig. 1. An overall taxonomy of topics on seq2seq models for neural abstractive text summarization.

sequence generation. In addition to a literature survey, we also implemented some of these methods in an open-source library, namely NATS (https://github.com/tshi04/NATS). Extensive set of experiments have been conducted on various benchmark text summarization datasets in order to examine the importance of different network components. The main contributions of this article can be summarized as follows:

- Provide a comprehensive literature survey of current advances of seq2seq models with an emphasis on the abstractive text summarization.
- Conduct a detailed review of the techniques used to tackle different challenges in RNN encoder-decoder architectures.
- Review different strategies for training seq2seq models and approaches for generating summaries.
- Provide an open-source library, which implements some of these models, and systematically investigate the effects of different network elements on the summarization performance.

The rest of this paper is organized as follows: An overall taxonomy of topics on seq2seq models for neural abstractive text summarization is shown in Figure 1. A comprehensive list of papers published till date on the topic of neural abstractive text summarization have been summarized in Tables 1 and 2. In Section 2, we introduce the basic seq2seq framework along with its extensions, including attention mechanism, pointing/copying mechanism, repetition handling, improving encoder or decoder, summarizing long documents and combining with extractive models. Section 3 summarizes different training strategies, including word-level training methods, such as cross-entropy training, and sentence-level training with RL algorithms. In Section 4, we discuss generating summaries using the beam search algorithm and other diverse beam decoding algorithms. In Section 5, we present details of our implementations and discuss our experimental results on the CNN/Daily Mail, Newsroom [44], and Bytecup (https://www.biendata.com/competition/bytecup2018/) datasets. We conclude this survey in Section 6.

2 THE RNN ENCODER-DECODER FRAMEWORK

In this section, we review different RNN-based encoder-decoder models for the neural abstractive text summarization. We will start with the basic seq2seq framework and attention mechanism. Then, we will describe more advanced network structures that can handle different challenges in

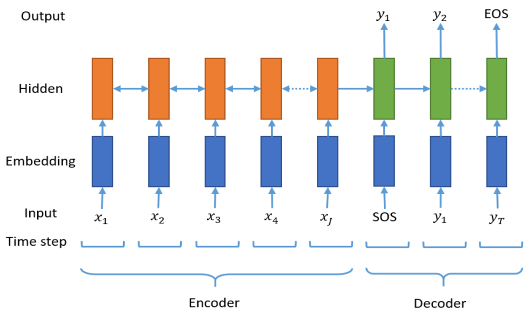

Fig. 2. The basic seq2seq model. SOS and EOS represent the start and end of a sequence, respectively.

the text summarization, such as repetition and out-of-vocabulary (OOV) words. We will highlight various existing problems and proposed solutions.

2.1 Seq2seq Framework Basics

A vanilla seq2seq framework for abstractive text summarization is composed of an encoder and a decoder. The encoder reads a source article, denoted by $x = (x_1, x_2, \ldots, x_J)$, and transforms it to hidden states $h^e = (h_1^e, h_2^e, \ldots, h_J^e)$; while the decoder takes these hidden states as the context input and outputs a summary $y = (y_1, y_2, \ldots, y_T)$. Here, x_i and y_j are one-hot representations of the tokens in the source article and summary, respectively. We use J and T to represent the number of tokens (document length) of the original source document and the summary, respectively. A summarization task is defined as inferring a summary y from a given source article x.

Encoders and decoders can be feed-forward networks, CNN [39, 40] or RNN. RNN architectures, especially long short-term memory (LSTM) [52] and gated recurrent unit (GRU) [21], have been most widely adopted for seq2seq models. Figure 2 shows a basic RNN seq2seq model with a bi-directional LSTM encoder and an LSTM decoder. The bi-directional LSTM is considered since it usually gives better document representations compared to a forward LSTM. The encoder reads a sequence of input tokens x and turns them into a sequences of hidden states $h = (h_1, h_2, h_3, \ldots, h_J)$. For the bi-directional LSTM, the input sequence is encoded as $\overrightarrow{h^e}$ and $\overleftarrow{h^e}$, where the right and left arrows denote the forward and backward temporal dependencies, respectively. Superscript e is the shortcut notation used to indicate that it is for the encoder. During the decoding, the decoder takes the encoded representations of the source article (i.e., hidden and cell states $\overrightarrow{h_J^e}, \overleftarrow{h_1^e}, \overrightarrow{c_J^e}, \overleftarrow{c_1^e}$) as the input and generates the summary y. In a simple encoder-decoder model, encoded vectors are used to initialize hidden and cell states of the LSTM decoder. For example, we can initialize them as follows:

$$h_0^d = \tanh\left(W_{\text{e2d}}\left(\overrightarrow{h}_J^e \oplus \overleftarrow{h}_1^e\right) + b_{\text{e2d}}\right), c_0^d = \overrightarrow{c_J^e} \oplus \overleftarrow{c_1^e} \tag{1}$$

Here, superscript d denotes the decoder and $\oplus$ is a concatenation operator. At each decoding step, we first update the hidden state h_t^d conditioned on the previous hidden states and input tokens, i.e., $h_t^d = \text{LSTM}(h_{t-1}^d, E_{y_{t-1}})$. Hereafter, we will not explicitly express the cell states in the input and output of LSTM, since only hidden states are passed to other parts of the model. Then, the vocabulary distribution can be calculated with $P_{\text{vocab},t} = \text{softmax}(W_{\text{d2v}} h_t^d + b_{\text{d2v}})$, where $P_{\text{vocab},t}$ is a vector whose dimension is the size of the vocabulary $\mathcal{V}$ and $\text{softmax}(v_t) = \frac{\exp(v_t)}{\sum_\tau \exp(v_\tau)}$ for each element v_t of a vector v. Therefore, the probability of generating the target token w in the vocabulary $\mathcal{V}$ is denoted as $P_{\text{vocab},t}(w)$.

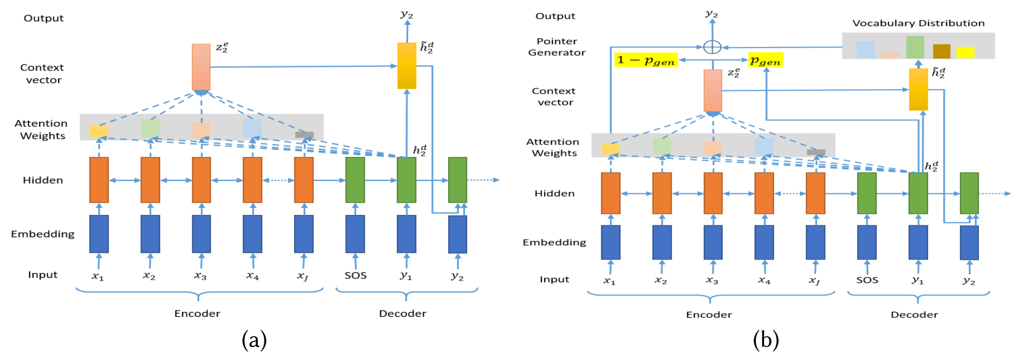

Fig. 3. (a) An attention-based seq2seq model. (b) The pointer-generator network.

This LSTM-based encoder-decoder framework was the foundation of many neural abstractive text summarization models [98, 109, 119]. However, there are many problems with this model. For example, the encoder is not well trained via back propagation through time [126, 141], since the paths from encoder to the output are relatively far apart, which limits the propagation of gradient signals. The accuracy and human-readability of generated summaries is also very low with a lot of OOV words[1] and repetitions. The rest of this section will discuss different models that were proposed in the literature to resolve these issues for producing better quality summaries.

2.2 Attention Mechanism

The attention mechanism has achieved great success and is commonly used in seq2seq models for different natural language processing (NLP) tasks [55], such as machine translation [3, 89], image captioning [147], and neural abstractive text summarization [98, 109, 119]. In an attention-based encoder-decoder architecture (shown in Figure 3(a)), the decoder not only takes the encoded representations (i.e., final hidden and cell states) of the source article as input, but also selectively focuses on parts of the article at each decoding step. For example, suppose we want to compress the source input "Kylian Mbappe scored two goals in four second-half minutes to send France into the World Cup quarter-finals with a thrilling 4-3 win over Argentina on Saturday." to its short version "France beat Argentina 4-3 to enter quarter-finals." When generating the token "beat", the decoder may need to attend "a thrilling 4-3 win" than other parts of the text. This attention can be achieved by an alignment mechanism [3], which first computes the attention distribution of the source tokens and then lets the decoder know where to attend to produce a target token. In the encoder-decoder framework depicted in Figures 2 and 3(a), given all the hidden states of the encoder, i.e., $h^e = (h_1^e, h_2^e, \ldots, h_J^e)$ and $h_j^e = \overrightarrow{h_j^e} \oplus \overleftarrow{h_j^e}$, and the current decoder hidden state h_t^d, the attention distribution α_t^e over the source tokens is calculated as follows:

$$\alpha_{tj}^e = \frac{\exp\left(s_{tj}^e\right)}{\sum_{k=1}^{J} \exp\left(s_{tk}^e\right)}, \qquad (2)$$

where the alignment score $s_{tj}^e = s(h_j^e, h_t^d)$ is obtained by the content-based score function, which has three alternatives as suggested in [89]:

[1]In the rest of this article, we will use <unk>, i.e., unknown words, to denote OOV words.

$$s\left(h_j^e, h_t^d\right) = \begin{cases} (h_j^e)^\top h_t^d & \text{dot} \\ (h_j^e)^\top W_{\text{align}} h_t^d & \text{general} \\ (v_{\text{align}})^\top \tanh\left(W_{\text{align}}\left(h_j^e \oplus h_t^d\right) + b_{\text{align}}\right) & \text{concat} \end{cases} \tag{3}$$

It should be noted that the number of additional parameters for "dot", "general", and "concat" approaches are 0, $|h_j^e| \times |h_t^d|$ and $((|h_j^e| + |h_t^d|) \times |v_{\text{align}}| + 2 \times |v_{\text{align}}|)$, respectively. Here $|\cdot|$ represents the dimension of a vector. The "general" and "concat" are commonly used score functions in the abstractive text summarization [109, 119]. One of the drawbacks of "dot" method is that it requires h_j^e and h_t^d to have the same dimension. With the attention distribution, we can naturally define the source side context vector for the target word as

$$z_t^e = \sum_{j=1}^{J} \alpha_{tj}^e h_j^e \tag{4}$$

Together with the current decoder hidden state h_t^d, we get the attention hidden state [89]

$$\tilde{h}_t^d = W_z\left(z_t^e \oplus h_t^d\right) + b_z. \tag{5}$$

Finally, the vocabulary distribution is calculated by

$$P_{\text{vocab},t} = \text{softmax}\left(W_{\text{d2v}} \tilde{h}_t^d + b_{\text{d2v}}\right) \tag{6}$$

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

$$h_{t+1}^d = \text{LSTM}\left(h_t^d, E_{y_t} \oplus \tilde{h}_t^d\right) \tag{7}$$

where the input is concatenation of E_{y_t} and $\tilde{h}_t^d$.

2.3 Pointing/Copying Mechanism

The pointing/copying mechanism [137] represents a class of approaches that generate target tokens by directly copying them from input sequences based on their attention weights. It can be naturally applied to the abstractive text summarization since summaries and articles can share the same vocabulary [119]. More importantly, it is capable of dealing with out-of-vocabulary (OOV) words [45, 46, 98, 119]. A variety of studies have shown a boost in performance after incorporating the pointing/copying mechanism into the seq2seq framework [15, 109, 119]. In this section, we review several alternatives of this mechanism for the abstractive text summarization.

2.3.1 Pointer Softmax [46]. The basic architecture of pointer softmax is described as follows. It consists of three fundamental components: short-list softmax, location softmax, and switching network. At decoding step t, a short-list softmax $P_{\text{vocab},t}$ calculated by Equation (6) is used to predict target tokens in the vocabulary. The location softmax gives locations of tokens that will be copied from the source article x to the target y_t based on attention weights α_t^e. With these two components, a switching network is designed to determine whether to predict a token from the vocabulary or copy one from the source article if it is an OOV token. The switching network is a multilayer perceptron (MLP) with a sigmoid activation function, which estimates the probability $p_{\text{gen},t}$ of generating tokens from the vocabulary based on the context vector z_t^e and hidden state h_t^d with

$$p_{\text{gen},t} = \sigma\left(W_{s,z} z_t^e + W_{s,h} h_t^d + b_s\right), \tag{8}$$

where $p_{\text{gen},t}$ is a scalar and $\sigma(a) = \frac{1}{1+\exp(-a)}$ is a sigmoid activation function. The final probability of producing the target token y_t is given by the concatenation of vectors $p_{\text{gen},t} P_{\text{vocab},t}$ and $(1 - p_{\text{gen},t})\alpha_t^e$.

2.3.2 Switching Generator-Pointer [98]. Similar to the switching network in pointer softmax [46], the switching generator-pointer is also equipped with a "switch", which determines whether to generate a token from the vocabulary or point to one in the source article at each decoding step. The switch is explicitly modeled by

$$p_{\text{gen},t} = \sigma\left(W_{s,z}z_t^e + W_{s,h}h_t^d + W_{s,E}E_{y_{t-1}} + b_s\right). \tag{9}$$

If the switch is turned on, the decoder produces a word from the vocabulary with the distribution $P_{\text{vocab},t}$ (see Equation (6)). Otherwise, the decoder generates a pointer based on the attention distribution α_t^e (see Equation (2)), i.e., $p_j = \arg\max_{j\in\{1,2,\ldots,J\}} \alpha_{tj}^e$, where p_j is the position of the token in the source article. When a pointer is activated, embedding of the pointed token E_{x_j} will be used as an input for the next decoding step.

2.3.3 CopyNet [45]. CopyNet has a differentiable network architecture and can be easily trained in an end-to-end manner. In this framework, the probability of generating a target token is a combination of the probabilities of two modes, i.e., generate-mode and copy-mode. First, CopyNet represents unique tokens in the vocabulary and source sequence by $\mathcal{V}$ and $\mathcal{X}$, respectively, and builds an extended vocabulary $\mathcal{V}_{\text{ext}} = \mathcal{V} \cup \mathcal{X} \cup \text{<unk>}$. Then, the vocabulary distribution over the extended vocabulary is calculated by

$$P_{\mathcal{V}_{\text{ext}}}(y_t) = P_g(y_t) + P_c(y_t), \tag{10}$$

where P_g and P_c are also defined on $\mathcal{V}_{\text{ext}}$, i.e.,

$$P_g(y_t) = \begin{cases} \frac{1}{Z}\exp\psi_g(y_t) & y_t \in \mathcal{V} \cup \text{<unk>} \\ 0 & \text{otherwise} \end{cases}, \quad P_c(y_t) = \begin{cases} \frac{1}{Z}\sum_{j:x_j=y_t}\exp\psi_c(x_j) & y_t \in \mathcal{X} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Here, Z is a normalization factor shared by both of the above equations. $\psi_g(y_t)$ is calculated with $\psi_g(y_t) = W_{\text{d2v}}\tilde{h}_t^d + b_{\text{d2v}}$ and $\psi_c(x_j)$ is an alignment score which can be calculated by Equation (3).

2.3.4 Pointer-Generator Network [119]. Pointer-generator network also has a differentiable network architecture (see Figure 3(b)). Similar to CopyNet [45], the vocabulary distribution over an extended vocabulary $\mathcal{V}_{\text{ext}}$ is calculated by

$$P_{\mathcal{V}_{\text{ext}}}(y_t) = p_{\text{gen},t}P_g(y_t) + (1 - p_{\text{gen},t})P_c(y_t), \tag{12}$$

where $p_{\text{gen},t}$ is obtained by Equation (9). Vocabulary distribution $P_g(y_t)$ and attention distribution $P_c(y_t)$ are defined as follows:

$$P_g(y_t) = \begin{cases} P_{\text{vocab},t}(y_t) & y_t \in \mathcal{V} \cup \text{<unk>} \\ 0 & \text{otherwise} \end{cases}, \quad P_c(y_t) = \begin{cases} \sum_{j:x_j=y_t}\alpha_{tj}^e & y_t \in \mathcal{X} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

The pointer-generator network has been used as the base model for many abstractive text summarization models (see Tables 1 and 2). Finally, it should be noted that $p_{\text{gen},t} \in (0, 1)$ in CopyNet and pointer-generator network can be viewed as a "soft-switch" to choose between generation and copying, which is different from "hard-switch" (i.e., $p_{\text{gen},t} = 0, 1$) in pointer softmax and switching generator-pointer [46, 98, 109].

2.4 Repetition Handling

One of the critical challenges for attention based seq2seq models is that the generated sequences have repetitions, since the attention mechanism tends to ignore the past alignment information [118, 131]. For summarization and headline generation tasks, model generated summaries suffer from both word-level and sentence-level repetitions. The latter is specific to summaries which consist of several sentences [98, 109, 119], such as those in CNN/Daily Mail dataset [98] and Newsroom

Table 1. An Overview of Different seq2seq Models for the Neural Abstractive Text Summarization (2015-2017)

Year	Reference	Highlights	Framework	Training	Optimizer	Datasets	Metrics
2015	Rush et al. [116]	Attention Based Summarization (ABS)	Bag-of-words, Convolution, Attention → Neural Network Language Model (NNLM)	XENT	SGD	DUC, Gigaword	ROUGE
	Lopyrev [88]	Simple Attention	LSTM → LSTM	XENT	RMSProp	Gigaword	BLEU
	Ranzato et al. [113]	Sequence-level Training	Elman, LSTM → Elman, LSTM	XENT, DAD, E2E, MIXER	SGD	Gigaword	ROUGE, BLEU
2016	Chopra et al. [22]	Recurrent Attentive Summarizer	Convolution Encoder, Attentive Encoder → Elman, LSTM	XENT	SGD	DUC, Gigaword	ROUGE
	Nallapati et al. [98]	Switch Generator-Pointer, Temporal-Attention, Hierarchical-Attention	RNN, Feature-rich Encoder → RNN	XENT	Adadelta	DUC, Gigaword, CNN/DM	ROUGE
	Miao and Blunsom [93]	Auto-encoding Sentence Compression, Forced-Attention Sentence Compression, Pointer Network	Encoder → Compressor → Decoder	XENT+RL	Adam	Gigaword	ROUGE
	Chen et al. [17]	Distraction	GRU → GRU	XENT	Adadelta	CNN, LCSTS	ROUGE
	Gulcehre et al. [46]	Pointer softmax	GRU → GRU	XENT	Adadelta	Gigaword	ROUGE
	Gu et al. [45]	CopyNet	GRU → GRU	XENT	SGD	LCSTS	ROUGE
	Zeng et al. [152]	Read-again, Copy Mechanism	LSTM/GRU/Hierarchical read-again encoder → LSTM	XENT	SGD	DUC, Gigaword	ROUGE
	Takase et al. [129]	Abstract Meaning Representation (AMR) based on ABS.	Attention-based AMR encoder → NNLM	XENT	SGD	DUC, Gigaword	ROUGE
2017	See et al. [119]	Pointer-Generator Network, Coverage	LSTM → LSTM	XENT	Adadelta	CNN/DM	ROUGE, METER
	Paulus et al. [109]	A Deep Reinforced Model, Intra-temporal and Intra-decoder Attention, Weight Sharing	LSTM → LSTM	XENT + RL	Adam	CNN/DM	ROUGE, Human
	Zhou et al. [159]	Selective Encoding, Abstractive Sentence Summarization	GRU → GRU	XENT	SGD	DUC, Gigaword, MSR-ATC	ROUGE
	Xia et al. [146]	Deliberation Networks	LSTM → LSTM	XENT	Adadelta	Gigaword	ROUGE
	Nema et al. [101]	Query-based, Diversity based Attention	GRU query encoder, document encoder → GRU	XENT	Adam	Debatepedia	ROUGE
	Tan et al. [130]	Graph-based Attention	Hierarchical Encoder → LSTM	XENT	Adam	CNN/DM, CNN, DailyMail	ROUGE
	Ling et al. [82]	Coarse-to-fine Attention	LSTM → LSTM	RL	SGD	CNN/DM	ROUGE, PPL
	Zhang et al. [155]	Sentence Simplification, Reinforcement Learning	LSTM → LSTM	RL	Adam	Newsela, WikiSmall, WikiLarge	BLEU, FKGL, SARI
	Li and Rush [79]	Deep Recurrent Generative Decoder (DRGD)	GRU → GRU, VAE	XENT, VAE	Adadelta	DUC, LCSTS	ROUGE
	Liu et al. [83]	Adversarial Training	Pointer-Generator Network	GAN	Adadelta	CNN/DM	ROUGE, Human
	Pasunuru et al. [108]	Multi-Task with Entailment Generation	LSTM document encoder and premise Encoder → LSTM Summary and Entailment Decoder	Hybrid-Objective	Adam	DUC, Gigaword, SNLI	ROUGE, METEOR, BLEU, CIDEr-D
	Gehring et al. [40]	Convolutional Seq2seq, Position Embeddings, Gated Linear Unit, Multi-step Attention	CNN → CNN	XENT	Adam	DUC, Gigaword	ROUGE
	Fan et al. [36]	Convolutional Seq2seq, Controllable	CNN → CNN	XENT	Adam	DUC, CNNDM	ROUGE, Human

Table 2. An Overview of Different seq2seq Models for the Neural Abstractive Text Summarization (2018)

Year	Reference	Highlights	Framework	Training	Optimizer	Datasets	Metrics
2018	Celikyilmaz et al. [15]	Deep Communicating Agents, Semantic Cohesion Loss	LSTM → LSTM	Hybrid-Objective	Adam	CNN/DM,NYT	ROUGE, Human
	Chen and Bansal [19]	Reinforce-Selected Sentence Rewriting	LSTM Encoder → Extractor → Abstractor	XENT + RL	SGD	DUC, CNN/DM	ROUGE, Human
	Hsu et al. [54]	Abstraction + Extraction, Inconsistency Loss	Extractor: GRU. Abstractor: Pointer-generator Network	Hybrid-Objective + RL	Adadelta	CNN/DM	ROUGE, Human
	Li et al. [78]	Actor-Critic	GRU → GRU	RL	Adadelta	DUC, CNN/DM, LCSTS	ROUGE
	Li et al. [73]	Abstraction + Extraction, Key Information Guide Network (KIGN)	KIGN: LSTM. Framework: Pointer-Generator Network	XENT	Adadelta	CNN/DM	ROUGE
	Lin et al. [81]	Global Encoding, Convolutional Gated Unit	LSTM → LSTM	XENT	Adam	Gigaword, LCSTS	ROUGE
	Pasunuru and Bansal [107]	Multi-Reward Optimization for RL: ROUGE, Saliency and Entailment.	LSTM → LSTM	RL	Adam	DUC, CNN/DM, SNLI, MultiNLI, SQuAD	ROUGE, Human
	Song et al. [125]	Structured-Infused Copy Mechanisms	Pointer-Generator Network	Hybrid-Objective	Adam	Gigaword	ROUGE, Human
	Cohan et al. [28]	Discourse Aware Attention	Hierarchical RNN LSTM Encoder → LSTM	XENT	Adagrad	PubMed, arXiv	ROUGE
	Guo et al. [47]	Multi-Task Summarization with Entailment and Question Generation	Multi-Task Encoder-Decoder Framework	Hybrid-Objective	Adam	DUC, Gigaword, SQuAD, SNLI	ROUGE, METEOR
	Cibils et al. [26]	Diverse Beam Search, Plagiarism and Extraction Scores	Pointer-Generator Network	XENT	Adagrad	CNN/DM	ROUGE
	Wang et al. [138]	Topic Aware Attention	CNN → CNN	RL	-	Gigaword, CNN/DM, LCSTS	ROUGE
	Kryściński et al. [71]	Improve Abstraction	LSTM Encoder → Decoder: Contextual Model and Language Model	XENT + RL	Asynchronous Gradient Descent Optimizer	CNN/DM	ROUGE, Novel n-gram Test, Human
	Gehrmann et al. [41]	Bottom-up Attention, Abstraction + Extraction	Pointer-Generator Network	Hybrid-Objective	Adagrad	CNN/DM, NYT	ROUGE, %Novel
	Zhang et al. [157]	Learning to Summarize Radiology Findings	Pointer-Generator Network + Background Encoder	XENT	Adam	Radiology Reports	ROUGE
	Jiang and Bansal [59]	Closed-book Training	Pointer-Generator Network + Closed-book Decoder	Hybrid-Objective + RL	Adam	DUC, CNN/DM	ROUGE, METEOR
	Chung et al. [25]	Main Pointer Generator	Pointer-Generator Network + Document Encoder	XENT	Adadelta	CNN/DM	ROUGE
	Chen et al. [18]	Iterative Text Summarization	GRU encoder, GRU decoder, iterative unit	Hybrid-Objective	Adam	DUC, CNN/DM	ROUGE

dataset [44]. In this section, we review several approaches that have been proposed to overcome the repetition problem.

2.4.1 Temporal Attention [98, 109]. Temporal attention method was originally proposed to deal with the attention deficiency problem in neural machine translation (NMT) [118]. Nallapati et al. [98] have found that it can also overcome the problem of repetition when generating multi-sentence summaries, since it prevents the model from attending the same parts of a source article by tracking the past attention weights. More formally, given the attention score s^e_{tj} in Equation (3), we can first define a temporal attention score as [109]:

$$s_{tj}^{\text{temp}} = \begin{cases} \exp\left(s_{tj}^{e}\right) & \text{if } t = 1 \\ \frac{\exp\left(s_{tj}^{e}\right)}{\sum_{k=1}^{t-1}\exp\left(s_{kj}^{e}\right)} & \text{otherwise} \end{cases} \tag{14}$$

Then, the attention distribution and context vector (see Equation (4)) are calculated with

$$\alpha_{tj}^{\text{temp}} = \frac{s_{tj}^{\text{temp}}}{\sum_{k=1}^{J} s_{tk}^{\text{temp}}}, \quad z_t^e = \sum_{j=1}^{J} \alpha_{tj}^{\text{temp}} h_j^e. \tag{15}$$

It can be seen from Equation (14) that, at each decoding step, the input tokens which have been highly attended will have a lower attention score via the normalization in time dimension. As a result, the decoder will not repeatedly attend the same part of the source article.

2.4.2 Intra-Decoder Attention [109]. Intra-decoder attention is another technique to handle the repetition problem for long-sequence generations. Compared to the regular attention based models, it allows a decoder to not only attend tokens in a source article but also keep track of the previously decoded tokens in a summary, so that the decoder will not repeatedly produce the same information.

For $t > 1$, intra-decoder attention scores, denoted by $s_{t\tau}^d$, can be calculated in the same manner as the attention score s_{tj}^e.[2] Then, the attention weight for each token is expressed as

$$\alpha_{t\tau}^{d} = \frac{\exp\left(s_{t\tau}^{d}\right)}{\sum_{k=1}^{t-1}\exp\left(s_{tk}^{d}\right)} \tag{16}$$

With attention distribution, we can calculate the decoder-side context vector by taking linear combination of the decoder hidden states, i.e., $h_{<t}^d$, as $z_t^d = \sum_{\tau=1}^{t-1} \alpha_{t\tau}^d h_\tau^d$. The decoder-side and encoder-side context vector will be both used to calculate the vocabulary distribution.

2.4.3 Coverage [119]. The coverage model was first proposed for the NMT task [131] to address the problems of the standard attention mechanism which tends to ignore the past alignment information. Recently, See et al. [119] introduced the coverage mechanism to the abstractive text summarization task. In their model, they first defined a coverage vector u_t^e as the sum of attention distributions of the previous decoding steps, i.e., $u_t^e = \sum_j^{t-1} \alpha_{tj}^e$. Thus, it contains the accumulated attention information on each token in the source article during the previous decoding steps. The coverage vector will then be used as an additional input to calculate the attention score

$$s_{tj}^e = \left(v_{\text{align}}\right)^\top \tanh\left(W_{\text{align}}\left(h_j^e \oplus h_t^d \oplus u_t^e\right) + b_{\text{align}}\right) \tag{17}$$

As a result, the attention at current decoding time-step is aware of the attention during the previous decoding steps. Moreover, they defined a novel coverage loss to ensure that the decoder does not repeatedly attend the same locations when generating multi-sentence summaries. Here, the coverage loss is defined as

$$\text{covloss}_t = \sum_j \min\left(\alpha_{tj}^e, u_{tj}^e\right), \tag{18}$$

which is upper bounded by 1.

[2]We have to replace h_j^e with h_τ^d in Equation (3), where $\tau \in \{1, \ldots, t-1\}$.

2.4.4 Distraction [17]. The coverage mechanism has also been used in [17] (known as *distraction*) for the document summarization task. In addition to the distraction mechanism over the attention, they also proposed a distraction mechanism over the encoder context vectors. Both mechanisms are used to prevent the model from attending certain regions of the source article repeatedly. Formally, given the context vector at current decoding step z_t^e and all historical context vectors $(z_1^e, z_2^e, \ldots, z_{t-1}^e)$ (see Equation (4)), the distracted context vector $z_t^{e,\text{dist}}$ is defined by $z_t^{e,\text{dist}} = \tanh(W_{\text{dist},z} z_t^e - W_{\text{hist},z} \sum_j^{t-1} z_j^e)$, where both $W_{\text{dist},z}$ and $W_{\text{hist},z}$ are diagonal parameter matrices.

2.5 Improving Encoded Representations

Although LSTM and bi-directional LSTM encoders[3] have been commonly used in the seq2seq models for the abstractive text summarization [98, 109, 119], representations of the source articles are still believed to be sub-optimal. In this section, we review some approaches that aim to improve the encoding process.

2.5.1 Selective Encoding [159]. The selective encoding model was proposed for the abstractive sentence summarization task [159]. Built upon an attention based encoder-decoder framework, it introduces a selective gate network into the encoder for the purpose of distilling salient information from source articles. A second layer representation, namely, distilled representation, of a source article is constructed over the representation of the first LSTM layer (a bi-directional GRU encoder in this work.). Formally, the distilled representation of each token in the source article is defined as

$$h_{\text{sel},j}^e = \text{gate}_{\text{sel},j} \times h_j^e \tag{19}$$

where $\text{gate}_{\text{sel},j}$ denotes the selective gate for token x_j and is calculated as follows:

$$\text{gate}_{\text{sel},j} = \sigma\left(W_{\text{sel},h} h_j^e + W_{\text{sel},\text{sen}} h_{\text{sen}}^e + b_{\text{sel}}\right) \tag{20}$$

where $h_{\text{sen}}^e = \overrightarrow{h_J^e} \oplus \overleftarrow{h_1^e}$. The distilled representations are then used for the decoding. Such a gate network can control information flow from an encoder to a decoder and can also select salient information, therefore, it boosts the performance of the sentence summarization task [159].

2.5.2 Read-Again Encoding [152]. Intuitively, read-again mechanism is motivated by human readers who read an article several times before writing a summary. To simulate this cognitive process, a read-again encoder reads a source article twice and outputs two-level representations. In the first read, an LSTM encodes tokens and the article as $(h_1^{e,1}, h_2^{e,1}, \ldots, h_J^{e,1})$ and $h_{\text{sen}}^{e,1} = h_J^{e,1}$, respectively. In the second read, we use another LSTM to encode the source text based on the outputs of the first read. Formally, the encoder hidden state of the second read $h_j^{e,2}$ is updated by

$$h_j^{e,2} = \text{LSTM}\left(h_{j-1}^{e,2}, E_{x_j} \oplus h_j^{e,1} \oplus h_{\text{sen}}^{e,1}\right) \tag{21}$$

The hidden states $(h_1^{e,2}, h_2^{e,2}, \ldots, h_J^{e,2})$ of the second read will be passed into decoders for summary generation.

2.6 Improving Decoder

2.6.1 Embedding Weight Sharing [109]. Sharing the embedding weights with the decoder is a practical approach that can boost the performance since it allows us to reuse the semantic and syntactic information in an embedding matrix during summary generation [57, 109]. Suppose the

[3]GRU and bi-directional GRU are also often seen in abstractive summarization papers.

embedding matrix is represented by W_{emb}, we can formulate the matrix used in the summary generation (see Equation (6)) as $W_{\text{d2v}} = \tanh(W_{\text{emb}}^{\top} \cdot W_{\text{proj}})$. By sharing model weights, the number of parameters is significantly less than a standard model since the number of parameters for W_{proj} is $|h_j^e| \times |h_t^d|$, while that for W_{d2v} is $|h_t^d| \times |\mathcal{V}|$, where $|h|$ represents the dimension of vector h and $|\mathcal{V}|$ denotes size of the vocabulary.

2.6.2 Deep Recurrent Generative Decoder (DRGD) [79]. Conventional encoder-decoder models calculate hidden states and attention weights in an entirely deterministic fashion, which limits the capability of representations and results in low-quality summaries. Incorporating variational auto-encoders (VAEs) [65, 115] into the encoder-decoder framework provides a practical solution for this problem. Inspired by the variational RNN proposed in [24] to model the highly structured sequential data, Li et al. [79] introduced a seq2seq model with DRGD that aims to capture latent structure information of summaries and improve the summarization quality. This model employs GRU as the basic recurrent model for both encoder and decoder. However, to be consistent with this survey paper, we will explain their ideas using LSTM instead.

There are two LSTM layers to calculate the decoder hidden state h_t^d. At the decoding step t, the first layer hidden state $h_t^{d,1}$ is updated by $h_t^{d,1} = \text{LSTM}^1(h_{t-1}^{d,1}, E_{y_{t-1}})$. Then, the attention weights α_{tj}^e and the context vector $z_t^{d,1}$ are calculated with the encoder hidden state h^e and the first layer decoder hidden state $h_t^{d,1}$ using Equations (2), (3), and (4). For the second layer, the hidden state $h_t^{d,2}$ is updated with $h_t^{d,2} = \text{LSTM}^2(h_{t-1}^{d,2}, E_{y_{t-1}} \oplus z_t^{d,1})$. Finally, the decoder hidden state is obtained by $h_t^d = h_t^{d,1} \oplus h_t^{d,2}$, where h^d is also referred to as the deterministic hidden state.

VAE is incorporated into the decoder to capture latent structure information of summaries which is represented by a multivariate Gaussian distribution. By using a reparameterization trick [33, 115], latent variables can be first expressed as $\xi_t = \mu_t + \eta_t \otimes \epsilon$, where the noise variable $\epsilon \sim \mathcal{N}(0, I)$, and Gaussian parameters μ_t and η_t in the network are calculated by

$$\mu_t = W_{\text{vae},\mu} h_t^{\text{enc}} + b_{\text{vae},\mu}, \log\left(\eta_t^2\right) = W_{\text{vae},\eta} h_t^{\text{enc}} + b_{\text{vae},\eta} \tag{22}$$

where h_t^{enc} is a hidden vector of the encoding process of the VAE and defined as

$$h_t^{\text{enc}} = \sigma\left(W_{\text{enc},\xi}\xi_{t-1} + W_{\text{enc},y}E_{y_{t-1}} + W_{\text{enc},h}h_{t-1}^d + b_{\text{enc}}\right). \tag{23}$$

With the latent structure variables ξ_t, the output hidden states h_t^{dec} can be formulated as

$$h_t^{\text{dec}} = \tanh\left(W_{\text{dec},\xi}\xi_t + W_{\text{dec},h}h_t^{d,2} + b_{\text{dec}}\right). \tag{24}$$

Finally, the vocabulary distribution is calculated by $P_{\text{vocab},t} = \text{softmax}(W_{\text{d2v}}h_t^{\text{dec}} + b_{\text{d2v}})$.

We primarily focused on the network structure of DRGD in this section. The details of VAE and its derivations can be found in [33], [65], [79], [115]. In DRGD, VAE is incorporated into the decoder of a seq2seq model, more recent works have also used VAE in the attention layer [6] and for the sentence compression task [93].

2.7 Summarizing Long Document

Compared to sentence summarization, the abstractive summarization for very long documents has been relatively less investigated. Recently, attention based seq2seq models with pointing/copying mechanism have shown their power in summarizing long documents with 400 and 800 tokens [109, 119]. However, performance improvement primarily attributes to copying and repetition/redundancy avoiding techniques [98, 109, 119]. For very long documents, we need to consider several important factors to generate high-quality summaries, such as saliency (i.e., significance), fluency, coherence and novelty [130]. Usually, seq2seq models combined with the beam search decoding algorithm can generate fluent and human-readable sentences. In this section, we review models that aim to improve the performance of long document summarization from the perspective of saliency.

Seq2seq models for long document summarization usually consists of an encoder with a hierarchical architecture which is used to capture the hierarchical structure of the source documents. The top-level salient information includes the important sentences [98, 130], chunks of texts [82], sections [28], and paragraphs [15], while the lower-level salient information represents keywords. Hereafter, we will use the term "chunk" to represent the top-level information. The hierarchical encoder first encodes tokens in a chunk for the chunk representation, and then, encodes different chunks in a document for the document representation. In this article, we only consider the single-layer forward LSTM[4] for both word and chunk encoders.

Suppose, the hidden states of chunk i and word j in this chunk are represented by h_i^{chk} and h_{ij}^{wd}. At decoding step t, we can calculate word-level attention weight $\alpha_{ij}^{\text{wd},t}$ for the current decoder hidden state h_t^d with $\alpha_{ij}^{\text{wd},t} = \frac{\exp(s_{ij}^{\text{wd},t})}{\sum_{k,l}\exp(s_{kl}^{\text{wd},t})}$ At the same time, we can also calculate chunk-level attention weight $\alpha_i^{\text{chk},t}$ by $\alpha_i^{\text{chk},t} = \frac{\exp(s_i^{\text{chk},t})}{\sum_k \exp(s_k^{\text{chk},t})}$, where both alignment scores $s_{ij}^{\text{wd},t} = s^{\text{wd}}(h_{ij}^{\text{wd}}, h_t^d)$ and $s_i^{\text{chk},t} = s^{\text{chk}}(h_i^{\text{chk}}, h_t^d)$ can be calculated using Equation (3). In this section, we will review four different models that are based on the hierarchical encoder for the task of long document text summarization.

2.7.1 Hierarchical Attention [98]. The intuition behind a hierarchical attention is that words in less important chunks should be less attended. Therefore, with chunk-level attention distribution $\alpha^{\text{chk},t}$ and word-level attention distribution $\alpha^{\text{wd},t}$, we first calculate re-scaled word-level attention distribution by

$$\alpha_{ij}^{\text{scale},t} = \frac{\alpha_i^{\text{chk},t}\alpha_{ij}^{\text{wd},t}}{\sum_{k,l}\alpha_k^{\text{chk},t}\alpha_{kl}^{\text{wd},t}} \tag{25}$$

This re-scaled attention will then be used to calculate the context vector using Equation (4), i.e., $z_t^e = \sum_{i,j}\alpha_{ij}^{\text{scale},t}h_{ij}^{\text{wd}}$. It should be noted that such hierarchical attention framework is different from the hierarchical attention network proposed in [150], where the chunk representation is obtained using $z_i^{\text{wd},t} = \sum_j \alpha_{ij}^{\text{wd},t}h_{ij}^{\text{wd}}$ instead of the last hidden state of the word-encoder.

2.7.2 Discourse-Aware Attention [28]. The idea of the discourse-aware attention is similar to that of the hierarchical attention-giving Equation (25). The main difference between these two attention models is that the re-scaled attention distribution in the discourse-aware attention is calculated by

$$\alpha_{ij}^{\text{scale},t} = \frac{\exp\left(\alpha_i^{\text{chk},t}s_{ij}^{\text{wd},t}\right)}{\sum_{k,l}\exp\left(\alpha_k^{\text{chk},t}s_{kl}^{\text{wd},t}\right)}, \tag{26}$$

where $s_{ij}^{\text{wd},t}$ is an alignment score instead of an attention weight.

2.7.3 Coarse-to-Fine Attention [82]. The coarse-to-fine (C2F) attention was proposed for computational efficiency. Similar to the hierarchical attention [98], the proposed model also has both chunk-level attention and word-level attention. However, instead of using word-level hidden states in all chunks for calculating the context vector, the C2F attention method first samples a

[4]The deep communicating agents model [15], which requires multiple layers of bi-directional LSTM, falls out of the scope of this survey.

chunk i from the chunk-level attention distribution, and then calculates the context vector using $z_t^e = \sum_j \alpha_{ij}^{\text{scale},t} h_{ij}^{\text{wd}}$. At the test time, the stochastic sampling of the chunks will be replaced by a greedy search.

2.7.4 Graph-Based Attention [130]. The aforementioned hierarchical attention mechanism implicitly captures the chunk-level salient information, where the importance of a chunk is determined solely by its attention weight. In contrast, the graph-based attention framework allows us to calculate the saliency scores explicitly using the pagerank algorithm [49, 104] on a graph whose vertices and edges are chunks of texts and their similarities, respectively. Formally, at the decoding time-step t, saliency scores for all input chunks are obtained by $f^t = (1-\lambda)(I - \lambda W^{\text{adj}}(t) D_{\text{adj}}^{-1}(t))^{-1} \chi_{\mathcal{T}}$, where adjacent matrix W^{adj} (similarity of chunks) is calculated by $W_{ij}^{\text{adj}} = h_i^{\text{chk}} W_{\text{par}} h_j^{\text{chk}}$. D_{adj} is a diagonal matrix with its (i,i)-element equal to the sum of the ith column of W^{adj}. λ is a damping factor. The vector $\chi_{\mathcal{T}}$ is defined as $\chi_{\mathcal{T},i} = \begin{cases} \frac{1}{|\mathcal{T}|} & i \in \mathcal{T} \\ 0 & \text{otherwise} \end{cases}$, where $\mathcal{T}$ is a topic (see [49] and [130] for more details). Finally, the graph-based attention distribution over a chunk can be obtained by

$$\alpha_i^{\text{chk},t} = \frac{\max\left(f_i^t - f_i^{t-1}, 0\right)}{\sum_k \left(\max\left(f_k^t - f_k^{t-1}, 0\right)\right)} \tag{27}$$

where f_i^0 is initialized with 0. It can be seen that the graph-based attention mechanism will focus on chunks that rank higher than the previous decoding step, i.e., $f_i^t > f_i^{t-1}$. Therefore, it provides an efficient way to select salient information from source documents.

2.8 Extraction + Abstraction

Extractive summarization approaches usually show a better performance comparing to the abstractive approaches [98, 119, 158] especially with respect to ROUGE measures. One of the advantages of the extractive approaches is that they can summarize source articles by extracting salient snippets and sentences directly from these documents [97], while abstractive approaches rely on word-level attention mechanism to determine the most relevant words to the target words at each decoding step. In this section, we review several studies that have attempted to improve the performance of the abstractive summarization by combining them with extractive models.

2.8.1 Extractor + Pointer-Generator Network [54]. This model proposes a unified framework that tries to leverage the sentence-level salient information from an extractive model and incorporate them into an abstractive model (a pointer-generator network). More formally, inspired by the hierarchical attention mechanism [98], they replaced the attention distribution α_t^e in the abstractive model with a scaled version α_t^{scale}, where the attention weights are expressed by $\alpha_{tj}^{\text{scale}} = \frac{\alpha_{tj}^{\text{extra}} \alpha_{tj}^{\text{wd}}}{\sum_k \alpha_{tk}^{\text{extra}} \alpha_{tk}^{\text{wd}}}$. Here, $\alpha_{tj}^{\text{extra}}$ is the sentence-level salient score of the sentence at word position j and decoding step t. Different from [98], the salient scores (sentence-level attention weights) are obtained from another deep neural network known as extractor [54].

During training, in addition to cross-entropy and coverage loss used in the pointer-generator network, this paper also proposed two other losses, i.e., *extractor loss* and *inconsistency loss*. The *extractor loss* is used to train the extractor and is defined by $L_{\text{ext}} = -\frac{1}{N} \sum_{n=1}^{N} g_n \log \beta_n + (1 - g_n) \log(1 - \beta_n)$, where g_n is the ground truth label for the nth sentence and N is the total number of sentences. The *inconsistency loss* is expressed as $L_{\text{inc}} = -\frac{1}{T} \sum_{t=1}^{T} \log(\frac{1}{|\mathcal{K}|} \sum_{j \in \mathcal{K}} \alpha_{tj}^e \alpha_{tj}^{\text{extra}})$, where $\mathcal{K}$ is the set of the top-k attended words and T is the total number of words in a summary. Intuitively, the inconsistency loss is used to ensure that the sentence-level attentions in the extractive model and word-level attentions in the abstractive model are consistent with each other. In other words, when word-level attention weights are high, the corresponding sentence-level attention weights should also be high.

2.8.2 Key-Information Guide Network (KIGN) [73]. This approach uses a guiding generation mechanism that leverages the key (salient) information, i.e., keywords, to guide decoding process. This is a two-step procedure. First, keywords are extracted from source articles using the TextRank algorithm [95]. Second, a KIGN encodes the key information and incorporates them into the decoder to guide the generation of summaries. Technically speaking, we can use a bi-directional LSTM to encode the key information and the output vector is the concatenation of hidden states, i.e., $h^{\text{key}} = \overrightarrow{h_N^{\text{key}}} \oplus \overleftarrow{h_1^{\text{key}}}$, where N is the length of the key information sequence. Then, the alignment mechanism is modified as $s_{tj}^e = (v_{\text{align}})^\top \tanh(W_{\text{align}}^e h_j^e + W_{\text{align}}^d h_t^d + W_{\text{align}}^{\text{key}} h^{\text{key}})$. Similarly, the soft-switch in the pointer-generator network is calculated using $p_{\text{gen},t} = \sigma(W_{s,z} z_t^e + W_{s,h} h_t^d + W_{s,\text{key}} h^{\text{key}} + b_s)$.

2.8.3 Reinforce-Selected Sentence Rewriting [19]. Most models introduced in this survey are built upon the encoder-decoder framework [98, 109, 119], in which the encoder reads source articles and turns them into vector representations, and the decoder takes the encoded vectors as input and generates summaries. Unlike these models, the reinforce-selected sentence rewriting model [19] consists of two seq2seq models. The first one is an extractive model (*extractor*) which is designed to extract salient sentences from a source article, while the second is an abstractive model (*abstractor*) which paraphrases and compresses the extracted sentences into a short summary. The abstractor network is a standard attention-based seq2seq model with the copying mechanism for handling OOV words. For the extractor network, an encoder first uses a CNN to encode tokens and obtains representations of sentences, and then it uses an LSTM to encode the sentences and represent a source document. With the sentence-level representations, the decoder (another LSTM) is designed to recurrently extract salient sentences from the document using the pointing mechanism [137]. This model has achieved the state-of-the-art performance on CNN/Daily Mail dataset and was demonstrated to be computationally more efficient than the pointer-generator network [119].

3 TRAINING STRATEGIES

In this section, we review different strategies to train the seq2seq models for abstractive text summarization. As discussed in [113], there are two categories of training methodologies, i.e., word-level and sequence-level training. The commonly used teacher forcing algorithm [8, 143] and cross-entropy training [9, 11] belong to the first category, while different RL-based algorithms [2, 113, 114] fall into the second. We now discuss the basic ideas of different training algorithms and their applications to seq2seq models for the text summarization. A comprehensive survey of deep RL for seq2seq models can be found in [62].

3.1 Word-Level Training

The word-level training for language models represents methodologies that try to optimize predictions of the next token [113]. For example, in the abstractive text summarization, given a source article x, a seq2seq model generates a summary y with the probability $P_\theta(y|x)$, where θ represents model parameters (e.g., weights W and bias b). In a neural language model [9], this probability can be expanded to

$$P_\theta(y|x) = \prod_{t=1}^{T} P_\theta(y_t|y_{<t}, x), \tag{28}$$

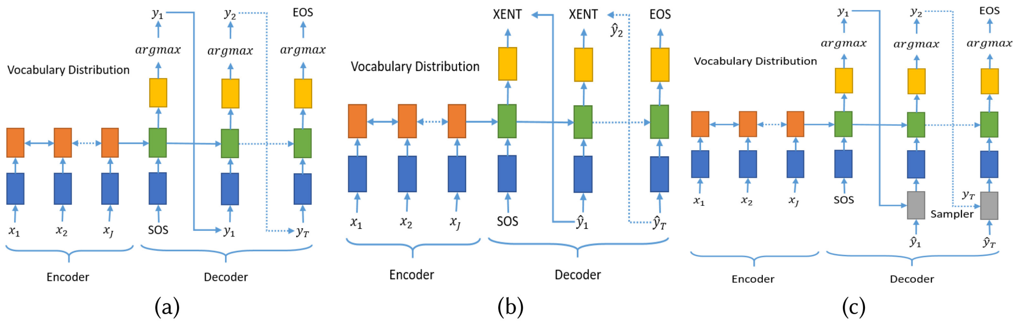

Fig. 4. (a) Generation process with a greedy search. (b) Training with the teacher-forcing algorithm. (c) Illustration of the scheduled sampling.

where each multiplier $P_\theta(y_t|y_{<t}, x)$, known as likelihood, is a conditional probability of the next token y_t given all previous ones denoted by $y_{<t} = (y_1, y_2, \ldots, y_{t-1})$. Intuitively, the text generation process can be described as follows: Starting with a special token " SOS" (start of sequence), the model generates a token y_t at a time t with the probability $P_\theta(y_t|y_{<t}, x) = P_{\text{vocab},t}(y_t)$. This token can be obtained by a sampling method or a greedy search, i.e., $y_t = \arg\max_{y_t} P_{\text{vocab},t}$ (see Figure 4(a)). The generated token will then be fed into the next decoding step. The generation is stopped when the model outputs " EOS" (end of sequence) token or when the length reaches a user-defined maximum threshold. In this section, we review different approaches for learning model parameters, i.e., θ. We will start with the commonly used end-to-end training approach, i.e., cross-entropy training, and then move on to two different methods for avoiding the problem of exposure bias.

3.1.1 *Cross-Entropy Training (XENT) [113].*

To learn model parameters θ, XENT maximizes the log-likelihood of observed sequences (ground-truth) $\hat{y}_t = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T)$, i.e.,

$$\log P_\theta(\hat{y}|x) = \sum_{t=1}^{T} \log P_\theta(\hat{y}_t|\hat{y}_{<t}, x) \tag{29}$$

which is equivalent to minimizing the cross entropy (XENT) loss, $\text{loss}_{\text{XENT}} = -\log P_\theta(\hat{y}|x)$. We show this training strategy in Figure 4(b). The algorithm is also known as the teacher-forcing algorithm [8, 143]. During training, it uses observed tokens (ground-truth) as input and aims to improve the probability of the next observed token at each decoding step. However, during testing, it relies on predicted tokens from the previous decoding step. This is the major difference between training and testing (see Figures 4(a) and (b)). Since the predicted tokens may not be the observed ones, this discrepancy will be accumulated over time and thus yields summaries that are very different from ground-truth summaries. This problem is known as exposure bias [8, 113, 134].

3.1.2 *Scheduled Sampling [8, 113, 134].*

Scheduled sampling algorithm, also known as Data As Demonstrator (DAD) [113, 134], has been proposed to solve the exposure bias problem. As shown in Figure 4(c), during training, the input at each decoding step comes from a sampler which can decide whether it is a model generated token y_t from the last step or an observed token $\hat{y}_t$ from training data. The sampling is based on a Bernoulli distribution

$$P_{\text{dad}}(y) = p_{\text{dad}}^{I(y=\hat{y}_t)} \cdot (1 - p_{\text{dad}})^{I(y=y_t)} \tag{30}$$

where p_{dad} is the probability of using a token from training data and $I(y = y_t)$ is a binary indicator function. In the scheduled sampling algorithm, p_{dad} is an annealing/scheduling function and decreases with training time from 1 to 0. As suggested by Bengio et al. [8], a scheduling function can take different forms, e.g.,

$$p_{\text{dad}} = \begin{cases} 1 - \alpha k & \text{linear decay} \\ \alpha^k & \text{exponential decay} \\ \frac{\alpha}{\alpha + \exp(k/\alpha)} & \text{inverse sigmoid decay} \end{cases} \tag{31}$$

where k is a training step and α is a parameter that guarantees $p_{\text{dad}} \in [0, 1]$. This strategy is often referred to as a *curriculum-learning algorithm* [8, 113, 155].

The main intuition behind this algorithm is that, at the beginning stage, a model with random parameters cannot generate relevant/correct tokens, therefore, a decoder takes ground-truth tokens from training data as input. As the training proceeds, the model gradually reduces the probability of using ground-truth tokens. By the end of the training, the model assumes that it has been well trained and can generate reasonable tokens; thus, the decoder can completely rely on its own predictions [8].

3.1.3 End-To-End Backprop (E2E) [113]. This algorithm is another method that exposes a model to its own predictions during training. At each decoding step, it still uses XENT to train the model parameters. However, the input is neither a ground truth token nor a model-generated token. Instead, it is a fusion of top-k tokens from the last decoding step, where k is a hyper-parameter. More specifically, the model first samples the top-k tokens, denoted as $(y_t^{\text{samp}_1}, y_t^{\text{samp}_2}, \ldots, y_t^{\text{samp}_k})$, from the vocabulary distribution $P_{\text{vocab},t}$. Then, it can re-scale their probabilities as follows:

$$P_{\text{samp},t}\left(y_t^{\text{samp}_i}\right) = \frac{P_{\text{vocab},t}\left(y_t^{\text{samp}_i}\right)}{\sum_j P_{\text{vocab},t}\left(y_t^{\text{samp}_j}\right)} \tag{32}$$

and obtain a vector in the embedding space by $E_{\text{samp},t} = \sum_{i=1}^{k} P_{\text{samp},t}(y_t^{\text{samp}_i}) E_{y_t^{\text{samp}_i}}$. This fused vector will be served as the input for the next decoding step. It should be noted that E2E also makes use of DAD in practice [113], where a sampler is used to determine whether to take fused vectors or embeddings of ground-truth tokens as input.

3.2 Sequence-Level Training

The sequence-level training with deep RL algorithms has recently received a lot of popularity in the area of neural abstractive text summarization [61, 62, 78, 107, 109, 113, 155] due to its ability to incorporate any user-defined metrics, including non-differentiable ROUGE scores, to train neural networks. In this section, we review several different policy-gradient algorithms that have been used to train abstractive text summarization models. For actor-critic algorithms and related work, the readers are encouraged to go through these publications [19, 62, 78].

In the RL setting, generating a sequence of tokens in a summary can be considered as a sequential decision-making process, where an encoder-decoder model is viewed as an agent, which first reads a source article x and initializes its internal state (hidden and cell states for LSTM). At the decoding step t, it updates the state and takes an action $y_t^\pi \in \mathcal{V}$ (i.e., picking a token from the vocabulary) according to a policy $\pi = P_\theta(y_t | y_{<t}, x)$. Here, the vocabulary is viewed as an action space. By the end of the decoding, it will produce a sequence of actions $(y_1^\pi, y_2^\pi, \ldots, y_T^\pi)$ and observe a reward $R(y_1^\pi, y_2^\pi, \ldots, y_T^\pi)$, which is usually ROUGE scores [80] in the context of text summarization. Then, RL algorithms will be used to update the agent by comparing the action sequence based on current policy with the optimal action sequence (i.e., the ground-truth summary). In this section, we will start with the commonly used REINFORCE for training seq2seq models. Then, we will introduce the MIXER algorithm (which can improve the convergence rate and

stability of the training) and the self-critic sequence training approach (which shows low variance for the gradient estimator).

3.2.1 REINFORCE [82, 113, 142, 155]. The goal of REINFORCE is to find parameters that maximize the expected rewards. Therefore, the loss function is defined as a negative expected reward, i.e., $L_\theta = -\mathbb{E}_{(y^\pi_{<T+1})\sim\pi_\theta} R(y^\pi_{<T+1})$, where $y^\pi_{<T+1}$ represents $y^\pi_1, y^\pi_2, \ldots, y^\pi_T$. The above equation can also be rewritten as

$$L_\theta = -\sum_{y^\pi_{<T+1}\in\mathcal{Y}} \pi_\theta\left(y^\pi_{<T+1}\right) R\left(y^\pi_{<T+1}\right) \tag{33}$$

where $\mathcal{Y}$ represents a set that contains all possible sequences. To optimize policy with respect to model parameters, we take the derivative of the loss function and obtain

$$\nabla_\theta L_\theta = -\sum_{y^\pi_{<T+1}\in\mathcal{Y}} \nabla_\theta \pi_\theta\left(y^\pi_{<T+1}\right) R\left(y^\pi_{<T+1}\right) = -\sum_{y^\pi_{<T+1}\in\mathcal{Y}} \pi_\theta\left(y^\pi_{<T+1}\right) \nabla_\theta \log \pi_\theta\left(y^\pi_{<T+1}\right) R\left(y^\pi_{<T+1}\right) \tag{34}$$

In the abstractive text summarization task, the policy is expressed as $\pi_\theta = P_\theta(y^\pi_{<T+1}|x)$ and according to Equation (28), the above equation can be expressed as follows:

$$\begin{aligned} \nabla_\theta L_\theta &= -\sum_{y^\pi_{<T+1}\in\mathcal{Y}} P_\theta\left(y^\pi_{<T+1}\middle|x\right) \cdot \left[\sum_{t=1}^{T} \nabla_\theta \log P_\theta\left(y^\pi_t\middle|y^\pi_{<t}, x\right)\right] \cdot R\left(y^\pi_{<T+1}\right) \\ &= -\mathbb{E}_{y^\pi_1\sim P_\theta(y^\pi_1|x)} \mathbb{E}_{y^\pi_2\sim P_\theta(y^\pi_2|y^\pi_1,x)} \cdots \mathbb{E}_{y^\pi_T\sim P_\theta(y^\pi_T|y^\pi_{<T},x)} \left[\sum_{t=1}^{T} \nabla_\theta \log P_\theta\left(y^\pi_t\middle|y^\pi_{<t}, x\right)\right] \cdot R\left(y^\pi_{<T+1}\right) \\ &= -\mathbb{E}_{y^\pi_{<T+1}\sim P_\theta(y^\pi_{<T+1}|x)} \left[\sum_{t=1}^{T} \nabla_\theta \log P_\theta\left(y^\pi_t\middle|y^\pi_{<t}, x\right)\right] \cdot R\left(y^\pi_{<T+1}\right) \end{aligned} \tag{35}$$

The reward $R(y^\pi_{<T+1})$ will be back-propagated to every node of the computational graph via the above gradient estimator. With the gradient, the model parameters are updated by

$$\theta \leftarrow \theta + \alpha \nabla_\theta L_\theta \tag{36}$$

where α is the learning rate.

As it can be seen from Equation (35), the computing gradient requires us to sample all sequences, which is not practical due to the presence of $|\mathcal{V}|^T$ possible number of sequences. Instead, REINFORCE approximates the expectation with a single sample, thus, the gradient is expressed as follows:

$$\nabla_\theta L_\theta \approx -\sum_{t=1}^{T} \nabla_\theta \log P_\theta\left(y^\pi_t\middle|y^\pi_{<t}, x\right) \cdot R\left(y^\pi_{<T+1}\right) \tag{37}$$

One of the problems associated with this method is high variance of gradient estimator, because it makes use of only one sample to train the model. A practical solution to alleviate this problem is introducing a baseline reward [114, 139, 147] denoted by b to the gradient, i.e.,

$$\nabla_\theta L_\theta = -\mathbb{E}_{y^\pi_{<T+1}\sim P_\theta(y^\pi_{<T+1}|x)} \left[\sum_{t=1}^{T} \nabla_\theta \log P_\theta\left(y^\pi_t\middle|y^\pi_{<t}, x\right)\right] \cdot \left(R\left(y^\pi_{<T+1}\right) - b\right) \tag{38}$$

The baseline b is arbitrary function but should not depend on $y^{\pi}_{<T+1}$ [114, 128]. In this way, it will not change the expectation of the gradient since $\mathbb{E}_{y^{\pi}_{<T+1} \sim P_\theta(y^{\pi}_{<T+1}|x)}[\sum_{t=1}^{T} \nabla_\theta \log P_\theta(y^{\pi}_t | y^{\pi}_{<t}, x)] \cdot b = 0$. The complete derivations of the above equation can be found in [114]. In practice, the gradient with the baseline is approximated with

$$\nabla_\theta L_\theta \approx -\left(R\left(y^{\pi}_{<T+1}\right) - b\right) \sum_{t=1}^{T} \nabla_\theta \log P_\theta\left(y^{\pi}_t \middle| y^{\pi}_{<t}, x\right) \tag{39}$$

Better ways of sampling a sequence and different approaches to calculate the baseline can be found in [62], [113], [147], and [151].

3.2.2 MIXER [113]. Training seq2seq models using REINFORCE may suffer from slow convergence and can also fail due to the large action space and poor initialization (which refers to randomly initialize parameters and start with random policy). To alleviate this problem, Ranzato et al. [113] modified REINFORCE by incorporating the idea of curriculum learning strategy and proposed a MIXER algorithm. In this algorithm, they first trained a seq2seq model for N-epochs to ensure RL starts with a better policy. Afterwards, in each batch and for each sequence, they used the cross entropy loss for the first $T - \Delta$ steps and REINFORCE for the remaining Δ steps, where Δ is an integer number. Training was continued for another N-epochs, where N is also an integer number. Then, they increased REINFORCE steps to 2Δ and continued training for another N-epochs. This process will repeat until the whole sequence is trained by REINFORCE. This algorithm has shown a better performance for greedy generation compared to XENT, DAD, and E2E in the task of abstractive text summarization.

3.2.3 Self-Critic Sequence Training (SCST) [15, 107, 109, 114]. The main idea of SCST is to use testing time inference algorithm as the baseline function in REINFORCE. Suppose the greedy search (see Figure 4(a)) is used to sample actions during testing. Then, at each training iteration, the model generates two action sequences, in which the first one $y^{\pi,\text{greedy}}_{<T+1}$ is from greedy search while the second one $y^{\pi}_{<T+1}$ is sampled from a distribution $P_\theta(y^{\pi}_{<T+1}|x)$. According to SCST, baseline b is defined as reward $R(y^{\pi,\text{greedy}}_{<T+1})$ to the first sequence. Therefore, the gradient of the loss function in SCST is expressed as

$$\nabla_\theta L_\theta \approx -\left(R\left(y^{\pi}_{<T+1}\right) - R\left(y^{\pi,\text{greedy}}_{<T+1}\right)\right) \sum_{t=1}^{T} \nabla_\theta \log P_\theta\left(y^{\pi}_t \middle| y^{\pi}_{<t}, x\right), \tag{40}$$

according to Equation (39). The SCST has shown low variance and can be effectively optimized with mini-batch SGD compared to REINFORCE [114]. It has also been demonstrated to be effective in improving the performance of seq2seq models for the task of abstractive text summarization [109]. In this work, the authors used the following RL loss to train their model:

$$L_{\text{RL}} \approx -\left(R\left(y^{\pi}_{<T+1}\right) - R\left(y^{\pi,\text{greedy}}_{<T+1}\right)\right) \sum_{t=1}^{T} \log P_\theta\left(y^{\pi}_t \middle| y^{\pi}_{<t}, x\right) \tag{41}$$

Although the model performs better than those trained with XENT in terms of ROUGE scores, human-readability of generated summaries is low. To alleviate this problem, the authors also defined a mixed loss function of RL and XENT, i.e., $L_{\text{MIXED}} = \gamma L_{\text{RL}} + (1-\gamma) L_{\text{XENT}}$, where $\gamma \in (0, 1)$ is a hyper-parameter. The model trained with the mixed loss can achieve better human-readability and ROUGE scores are still better than those obtained with XENT. They also used scheduled sampling to reducing exposure bias, in which the scheduling function is a constant ($p_{\text{dad}} = 0.75$).

We have reviewed different RNN encoder-decoder architectures and training strategies in the last two sections. Now, we are at the position to generate summaries for given source articles.

4 SUMMARY GENERATION

Generally speaking, the goal of summary generation is to find an optimal sequence $y^*_{<T+1}$ such that

$$y^*_{<T+1} = \arg\max_{y_{<T+1} \in \mathcal{Y}} \log P_\theta(y_{<T+1}|x) = \arg\max_{y_{<T+1} \in \mathcal{Y}} \sum_{t=1}^{T} \log P_\theta(y_t|y_{<t}, x) \tag{42}$$

where $\mathcal{Y}$ represents a set that contains all possible sequences (summaries). However, since it has $|\mathcal{V}|^T$ elements, the exact inference is intractable in practice [116]. Here, $\mathcal{V}$ represents the output vocabulary. In this section, we review the beam search algorithm and its extensions for approximating the exact inference.

4.1 Greedy and Beam Search

As shown in Figure 4(a), we can generate a sub-optimal sequence with greedy search, i.e.,

$$y^*_t = \arg\max_{y_t \in \mathcal{V}} \log P_\theta(y_t|y_{<t}, x) \tag{43}$$

at each decoding step t. Although greedy search is computationally efficient, human-readability of generated summaries is low.

Beam search algorithm is a compromise between greedy search and exact inference and has been commonly employed in different language generation tasks [89, 116, 119]. Beam search is a graph-search algorithm that generates sequences from left to right by retaining only B top scoring (top-B) sequence-fragments at each decoding step. More formally, we denote decoded top-B sequence fragments, also known as hypotheses [116], at time-step $t-1$ as $y_{<t,1}, y_{<t,2}, \ldots, y_{<t,B}$ and their scores as $S^{\text{bm}}_{<t,1}, S^{\text{bm}}_{<t,2}, \ldots, S^{\text{bm}}_{<t,B}$. For each fragment $y_{<t,b}$, we first calculate $P_\theta(y^{\text{cand}}_{t,b}|y_{<t,b}, x)$, which determines B most probable words $y^{\text{cand}}_{t,b,1}, y^{\text{cand}}_{t,b,2}, \ldots, y^{\text{cand}}_{t,b,B}$ to expand it. The score for each expanded fragment, i.e., new hypotheses, $y^{\text{cand}}_{<t+1,b,b'}$ can then be updated with either

$$S^{\text{cand}}_{t,b,b'} = S^{\text{bm}}_{<t,b} \times P_\theta\left(y^{\text{cand}}_{t,b,b'} \middle| y_{<t,b}, x\right) \tag{44}$$

where $S^{\text{bm}}_{<t,b}$ is initialized with 1, or $S^{\text{cand}}_{t,b,b'} = S^{\text{bm}}_{<t,b} + \log P_\theta(y^{\text{cand}}_{t,b,b'}|y_{<t,b}, x)$, where $S^{\text{bm}}_{<t,b}$ is initialized with 0. Here, b and b' are labels of a current hypothesis and a word candidate, respectively. This yields $B \times B$ expanded fragments, i.e., new hypotheses, in which only the top-B of them along with their scores are retained for the next decoding step. This procedure will be repeated until" "EOS" token is generated. In Algorithm 1, we show pseudo-codes of a beam search algorithm for generating summaries with Seq2Seq models given the beam size of B and batch size of 1.

4.2 Diversity-Promoting Algorithms

Despite widespread applications, beam search algorithm suffered from lacking of diversity within a beam [42, 68, 76, 136]. In other words, the top-B hypotheses may differ by just a couple tokens at the end of sequences, which not only limits applications of summarization systems but also wastes computational resources [74, 136]. Therefore, it is important to promote the diversity of generated sequences. Ippolito et al. [58] performed an extensive analysis of different post-training decoding algorithms that aim to increase the diversity during decoding. They have also shown the power of oversampling (i.e., first sampling additional candidates, and then, filtering them to the desired number) in improving the diversity without sacrificing performance. In this section, we briefly introduce some studies that aim to increase the diversity of the beam search algorithm for abstractive summarization models.

ALGORITHM 1: Beam search algorithm for decoding seq2seq models.

```
Input: Source article $x$, beam size $B$, summary length $T$, model parameters $\theta$;
Output: $B$-best summaries;
Initialize:
Output sequences $Q^{\text{seq}} = [\text{SOS}]_{B\times T}$;
Accumulated probabilities $Q^{\text{prob}} = [1.0]_{B\times 1}$;
The last decoded tokens $Q^{\text{word}} = [\text{SOS}]_{B\times 1}$;
States (hidden and cell states for LSTM) $Q^{\text{states}} = [0.0]_{B\times |h_t^d|}$;
Context vectors $Q^{\text{ctx}} = [0.0]_{B\times |z_t^e|}$;
Compute $(h_1^e, h_2^e, \dots, h_J^e)$ with encoder;
Update $Q^{\text{states}}$ with encoder states;
for $t$=1, $T$ do
    Initialize candidates $Q^{\text{cand, seq}}$, $Q^{\text{cand, prob}}$, $Q^{\text{cand, word}}$, $Q^{\text{cand, states}}$, $Q^{\text{cand, ctx}}$ by repeating $Q^{\text{seq}}$,
    $Q^{\text{prob}}$, $Q^{\text{word}}$, $Q^{\text{states}}$ and $Q^{\text{ctx}}$ $B$ times, respectively;
    for $b$=1, $B$ do
        Compute $P_\theta(y_{t,b}^{\text{cand}}|y_{<t,b}, x)$ using decoder LSTM cell with input $(h_1^e, h_2^e, \dots, h_J^e)$, $Q_b^{\text{word}}$,
        $Q_b^{\text{states}}$ and $Q_b^{\text{ctx}}$;
        Select the top-$B$ candidate words $y_{t,b,b'}^{\text{cand}}$, where $b' = 1, 2, \dots, B$;
        Select corresponding probability $P_\theta(y_{t,b,b'}^{\text{cand}}|y_{<t,b}, x)$, hidden states $h_{t,b,b'}^d$, cell states $c_{t,b,b'}^d$
            and context vector $z_{t,b,b'}^e$;
        Update elements of $Q_{b',b,t}^{\text{cand, seq}}$, $Q_{b',b}^{\text{cand, word}}$ with $y_{t,b,b'}^{\text{cand}}$;
        Update elements of $Q_{b',b}^{\text{cand, states}}$ with $h_{t,b,b'}^d$ and $c_{t,b,b'}^d$;
        Update elements of $Q_{b',b}^{\text{cand, ctx}}$ with $z_{t,b,b'}^e$;
        Update $Q_{b',b}^{\text{cand, prob}}$ with Equation (44);
    end
    Flatten $Q^{\text{cand, prob}}$ and choose $B$ best hypotheses;
    Update $Q_t^{\text{seq}}$, $Q^{\text{prob}}$, $Q^{\text{word}}$, $Q^{\text{states}}$, $Q^{\text{ctx}}$ with corresponding candidates.
end
```

4.2.1 Maximum Mutual Information (MMI) [74–76]. The MMI-based methods were originally proposed for neural conversation models and then applied to other tasks, such as machine translation and summarization [75, 76]. The basic intuition here is that a desired model should not only take into account the dependency of a target on a source, but also should consider the likelihood of the source for a given target, which is achieved by replacing the log-likelihood of the target, i.e., $\log P_\theta(y|x)$ in Equation (42), with pairwise mutual information of the source and target, defined by $\log \frac{P_\theta(y,x)}{P_\theta(x)P_\theta(y)}$. During the training, model parameters are learned by maximizing mutual information. When generating sequences, the objective is expressed as follows:

$$y^* = \arg\max_{y\in\mathcal{Y}} \log \frac{P_\theta(y,x)}{P_\theta(x)P_\theta(y)} = \arg\max_{y\in\mathcal{Y}} (\log P_\theta(y|x) - \log P_\theta(y)) \tag{45}$$

However, it is obvious that calculating $P_\theta(y)$ is intractable. Thus, several approximation methods have been proposed in the literature to alleviate this problem [74, 75]. These approximation method can be summarized by the following three steps:

- Train two seq2seq models, one for $P_{\theta_1}(y|x)$ and the other for $P_{\theta_2}(x|y)$, where θ_1 and θ_2 are the model parameters.
- Generate a diverse N-best list of sequences based on $P_{\theta_1}(y|x)$. To achieve this goal, the method for calculating the scores for beam search algorithm has been modified as

$$S^{\text{beam}}_{k,k'} = b_{<t,k} + \log P_\theta(y_{t,k'}|y_{<t,k}, x) - \gamma k'. \tag{46}$$

  By adding the last term $\gamma k'$, the model explicitly encourages hypotheses from different parents, i.e., different k, which results in more diverse results. Therefore, parameter γ is also known as the diversity rate which indicates the degree of diversity integrated into beam search algorithm [76].
- Re-rank N-best list by linearly combining $P_{\theta_1}(y|x)$ and $P_{\theta_2}(x|y)$. The ranking score for each candidate in the N-best list is defined as follows:

$$S_{\text{rank}}(y) = \log P_{\theta_1}(y|x) + \lambda \log P_{\theta_2}(x|y) + \beta\Omega(y), \tag{47}$$

  where $\Omega(y)$ is a task-specific auxiliary term. λ and β are parameters that can be learned using minimum error rate training [103] on the development dataset.

4.2.2 Diverse Beam Search (DBS) [26, 136]. DBS is another approach that aims to increase the diversity of standard beam search algorithm. It first partitions the hypotheses into G groups. Then, at each decoding step, it sequentially performs a beam search on each group based on a dissimilarity augmented scoring function

$$S^{\text{cand}}_{t,b,b'} = S^{\text{cand}}_{<t,b} + \log P_\theta\left(y^{\text{cand},g}_{t,b,b'}\middle|y^{g}_{<t,b}, x\right) + \lambda_g \Delta\left(y^{\text{cand},g}_{<t+1,b,b'}; y^{1}_{<t+1}, \ldots, y^{g-1}_{<t+1}\right), \tag{48}$$

where $\lambda_g \geq 0$ is a parameter. $\Delta(y^{\text{cand},g}_{<t+1,b,b'}; y^{1}_{<t+1}, \ldots, y^{g-1}_{<t+1})$ represents a diversity function, which measures the dissimilarity or distance between candidate $y^{\text{cand},g}_{<t+1,b,b'}$ in group g and sequences in groups from 1 to $g-1$. Standard beam search is applied to group 1. Intuitively, the generated sequences in different groups are very different from each other due to the penalty of diversity functions. In [26], DBS has been combined with pointer-generator network [119] to improve the diversity of the model produced summaries.

5 IMPLEMENTATIONS AND EXPERIMENTS

Apart from a comprehensive literature survey and a detailed review of different techniques for network structures, training strategies and summary generations, we have also developed an open-source library, namely, NATS (https://github.com/tshi04/NATS), based on RNN seq2seq framework for abstractive text summarization [124]. In this section, we first introduce the details of our implementations and then systematically experiment with different network elements and hyper-parameters on three public available datasets, i.e., CNN/Daily Mail, Newsroom, and Bytecup.

5.1 Implementations

The NATS is equipped with following important features:

- **Attention-based seq2seq framework.** We implemented the attention-based seq2seq model shown in Figure 3(a). The encoder and decoder can be chosen to be either LSTM or GRU. The attention scores can be calculated with one of three alignment methods given in Equation (3).
- **Pointer-generator network.** Based on the attention-based seq2seq framework, we implemented pointer-generator network discussed in Section 2.3.4.

Table 3. Basic Statistics of the CNN/Daily Mail Dataset

	CNN/Daily Mail			Newsroom			Bytecup		
	Train	**Dev**	**Test**	**Train**	**Dev**	**Test**	**Train**	**Dev**	**Test**
# pairs	287,227	13,368	11,490	992,985	108,612	108,655	892,734	111,592	111,592
Article Length	751	769	778	773	766	765	640	640	639
Headline Length	-	-	-	10	10	10	12	12	12
Summary Length	55	61	58	31	31	31	-	-	-

- **Intra-temporal attention mechanism.** The temporal attention can work with all three alignment methods.
- **Intra-decoder attention mechanism.** The alignment method for intra-decoder attention is the same as that for the attention mechanism.
- **Coverage mechanism.** To handle the repetition problem, we implemented the coverage mechanism discussed in Section 2.4.3. If coverage is switched off, coverage loss will be set to 0.
- **Weight-sharing mechanism.** As discussed in Section 2.6.1, weight-sharing mechanism can boost the performance using significantly fewer parameters.
- **Beam-search algorithm.** We implemented an efficient beam-search algorithm that can also handle the case when the batch size > 1.
- **Unknown words replacement.** Similar to [17], we implemented a heuristic unknown words replacement technique to boost the performance. Theoretically, a pointer-generator network may generate OOV words even with the copying mechanism, because <unk> is still in the extended vocabulary. Thus, after the decoding is completed, we manually check <unk> in summaries and replace them with words in source articles using attention weights. This meta-algorithm can be used for any attention-based seq2seq model.

5.2 Datasets

5.2.1 CNN/Daily Mail Dataset. CNN/Daily Mail dataset (https://github.com/abisee/cnn-dailymail) consists of more than 300K news articles and each of them is paired with several highlights, known as multi-sentence summaries [98, 119]. We have summarized the basic statistics of the dataset in Table 3. There are primarily two versions of this dataset. The first version anonymizes name entities [98], while the second one keeps the original texts [119]. In this article, we used the second version and obtained processed data from See et al. [119] (https://github.com/JafferWilson/Process-Data-of-CNN-DailyMail).

5.2.2 Newsroom Dataset. The Cornell Newsroom dataset (https://summari.es/) [44] was recently released and consists of 1.3 million article-summary pairs, out of which 1.2 million of them are publicly available for training and evaluating summarization systems. We first used newsroom library (https://github.com/clic-lab/newsroom) to scrape and extract the raw data. Then, texts were tokenized with SpaCy package. We developed a data processing tool to tokenize texts and prepare input for NATS. In this survey, we created two datasets for text summarization and headline generation, respectively. Their basic statistics are shown in Table 3.

5.2.3 Bytecup Dataset. Byte Cup 2018 International Machine Learning Contest (https://www.biendata.com/competition/bytecup2018/) released a new dataset, which will be referred to as Bytecup dataset in this survey, for the headline-generation task. It consists of 1.3 million pieces of articles, out of which 1.1 million are released for training. In our experiments, we create training, development, and testing sets (0.8/0.1/0.1) based on this training dataset. Texts are tokenized using Stanford CoreNLP package and prepared with our data processing tool. The basic statistics of the dataset are shown in Table 3.

5.3 Parameter Settings

In all our experiments, we set the dimension of word embeddings and hidden states (for both encoder and decoder) as 128 and 256, respectively. During training, the embeddings are learned from scratch. Adam [64] with hyper-parameter $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ is used for stochastic optimization. Learning rate is fixed to 0.0001 and mini-batches of size 16 are used. Gradient clipping is also used with a maximum gradient norm of 2.0. For all datasets, the vocabulary consists of $50K$ words and is shared between source and target. For the CNN/Daily Mail dataset, we truncate source articles to 400 tokens and limit the length of summaries to 100 tokens. For the Newsroom dataset, source articles, summaries, and headlines are truncated to 400, 50, and 20, respectively. For the Bytecup dataset, lengths of source articles and headlines are also limited to 400 and 20 tokens, respectively. During training, we run 35 epochs for the CNN/Daily Mail dataset and 20 epochs for the Newsroom and Bytecup datasets. During testing, we set the size of a beam to 5.

5.4 ROUGE Evaluations

Recall-Oriented Understudy for Gisting Evaluation (ROUGE) scores were first introduced in [80] and have become standard metrics for evaluating abstractive text summarization models. They determine the quality of summarization by counting the number of overlapping units (i.e., *n*-grams, word sequences, and word pairs) between machine-generated and golden-standard (human-written) summaries [80]. Within all different ROUGE measures, ROUGE-1 (unigram), ROUGE-2 (bigram), and ROUGE-L (longest common subsequence) have been most widely used for single-document abstractive summarization [119]. In this article, different models are evaluated using the pyrouge (https://pypi.python.org/pypi/pyrouge/0.1.0) package, which provides precision, recall, and F-score for these measures.

5.5 Experiments on CNN/Daily Mail Dataset

In the past few years, the CNN/Daily Mail dataset has become a standard benchmark dataset used for evaluating the performance of different summarization models that can generate multi-sentence summaries for relatively longer documents [97, 98, 109, 119, 156]. In our experiments, we systematically investigated the effects of six network components in the seq2seq framework on summarization performance, including (i) alignment methods in the attention mechanism, (ii) pointing mechanism, (iii) intra-temporal attention, (iv) intra-decoder attention, (v) weight sharing, and (vi) coverage mechanism.

Our experimental results are shown in Table 4. To effectively represent different models, ID of each model consists of a letter followed by five binary-indicators, corresponding to the six important components. The letters "G", "D", and "C" denote alignment methods "general", "dot", and "concat", respectively. 1 and 0 indicates if a component is switched on or off, respectively. At first, it can be clearly seen that the performance of three basic attention-based models (i.e., G00000, D00000, C00000) are close to each other. In these tests, we still keep the OOV tokens in generated summaries when performing the ROUGE evaluations, which results in relatively lower ROUGE precision scores. Therefore, ROUGE F-scores may be lower than those reported in the literature [98, 119]. Comparing G10000 with G00000, and C10000 with C00000, we find that pointing mechanism significantly improves the performance of attention-based seq2seq models. By analyzing summaries, we observed that most of the tokens are copied from source articles, which results in

Table 4. ROUGE Scores on the CNN/Daily Mail Dataset in Our Experiments

Model ID	Attention	Pointer-generator	Intra-temporal	Intra-decoder	Weight sharing	Coverage	R-1	R-2	R-L
G00000	general	-	-	-	-	-	27.62	10.15	25.81
G10000	general	✓	-	-	-	-	33.85	14.08	31.33
G11000	general	✓	✓	-	-	-	36.78	15.82	34.05
G11010	general	✓	✓	-	✓	-	37.07	15.93	34.22
G11100	general	✓	✓	✓	-	-	36.74	15.78	33.98
G11110	general	✓	✓	✓	✓	-	**37.60**	**16.27**	**34.77**
D00000	dot	-	-	-	-	-	27.60	10.19	25.81
D11100	dot	✓	✓	✓	-	-	37.00	15.87	34.24
D11110	dot	✓	✓	✓	✓	-	**37.65**	**16.25**	**34.85**
C00000	concat	-	-	-	-	-	27.61	10.16	25.78
C10000	concat	✓	-	-	-	-	35.55	15.18	32.84
C10001	concat	✓	-	-	-	✓	38.64	16.70	35.63
C10100	concat	✓	-	✓	-	-	36.51	15.75	33.70
C10101	concat	✓	-	✓	-	✓	**39.23**	**17.28**	36.02
C10110	concat	✓	-	✓	✓	-	36.46	15.68	33.69
C10111	concat	✓	-	✓	✓	✓	39.14	17.13	**36.04**

summaries that are similar to the ones generated by the extractive models.[5] As discussed in Section 2.3.4, another advantage of pointing mechanism is that it can effectively handle OOV tokens.

The remaining four components are tested upon the pointer-generator network. By comparing G11000 and G10000, we see that the intra-temporal attention increases almost 3 ROUGE points. This might be because of its capability of reducing repetitions. However, most of our models that combine intra-temporal attention with "concat" failed during training after a few epochs. Thus, we did not report these results. As to intra-decoder attention, we observe from G11000, G11010, G11100, and G11110 that it does not boost the performance of the model before adding weight-sharing mechanism. However, in the case of "concat", the models with intra-decoder attention have a better performance. Weight sharing mechanism does not always boost the performance of the models (according to the comparison of C10100 and C10110). However, as aforementioned, models that adopt weight sharing mechanism have much fewer parameters. Finally, we find the coverage mechanism can significantly boost performance by at least 2 ROUGE points, which is consistent with the results presented in [119]. It should be noted that the coverage mechanism can only work with the "concat" attention mechanism according to Section 2.4.3.

5.6 Experiments on Newsroom and Bytecup Datasets

We also tested NATS toolkit on the Newsroom dataset, which was released recently. In our experiments, we tokenized the raw data with three different packages and generated three versions of the dataset for the task of text summarization and three versions for the task of headline generation. Experimental results obtained with the models G11110 and C10110 on the released testing set [44] are shown in Table 5. It can be observed that G11110 performs better than C10110 on CNN/Daily Mail data from Table 4, however, C10110 achieves better ROUGE scores in both text summarization and headline generation tasks on Newsroom dataset. Finally, we summarize our results for

[5]The extractive models attempt to extract sentences from the source articles.

Table 5. ROUGE Scores on Newsroom and Bytecup Datasets

	Newsroom-Summary			Newsroom-Title			Bytecup		
Model	**R-1**	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
G11110	39.11	27.54	35.99	25.23	11.46	23.72	39.04	22.72	35.90
C10110	39.36	27.86	36.35	26.56	12.31	25.04	39.13	22.98	36.14

the Bytecup headline generation dataset in Table 5. C10110 achieves slightly better scores than G11110.

6 CONCLUSION AND FUTURE DIRECTIONS

Being one of the most successful applications of seq2seq models, neural abstractive text summarization has become a prominent research topic that has gained a lot of attention from both industry and academia. In this article, we provided a comprehensive survey on the recent advances of seq2seq models for the task of abstractive text summarization. This work primarily focuses on the challenges associated with neural network architectures, model parameter inference mechanisms and summary generation procedures, and the solutions of different models and algorithms. We also provided a taxonomy of these topics and an overview of different seq2seq models for the abstractive text summarization. As part of this survey, we developed an open source toolkit, namely, NATS, which is equipped with several important features, including attention, pointing mechanism, repetition handling, and beam search. In our experiments, we first summarized the experimental results of different seq2seq models in the literature on the widely used CNN/Daily Mail dataset. We also conducted extensive experiments on this dataset using NATS to examine the effectiveness of different neural network components. Finally, we established benchmarks for two recently released datasets, i.e., Newsroom and Bytecup.

Despite advances of Seq2Seq models in the abstractive text summarization task, there are still many research challenges that are worth pursing in the future.

(1) *Large Transformers.* These large-scale models are first pre-trained on massive text corpora with self-supervised objectives and then fine-tuned on downstream tasks. They have achieved state-of-the-art performance on a variety of summarization benchmark datasets [34, 72, 85, 112, 148, 153]. In [153], the pre-trained encoder-decoder model can outperform previous state-of-the-art results on several datasets by fine-tuning with limited supervised examples, which shows that pre-trained models are promising candidates for zero-shot and low-resource summarization.

(2) *Reinforcement Learning (RL).* RL-based training strategies can incorporate any user-defined metrics, including non-differentiable ones, as rewards to train summarization models [56, 62, 107, 109]. These metrics can be ROUGE [80], BERTScore [154], or saliency and entailment rewards [107] inferred from the Natural Language Inference task [13]. Therefore, we may improve current models with RL by leveraging external resources and characteristics of different datasets.

(3) *Summary Generation.* Most Seq2Seq-based summarization models rely on beam-search algorithm to generate summaries. Recently, sampling-based approaches have achieved success in open-ended language generation [53] since they can increase the diversity of the generated texts. It is a promising research direction that can potentially increase the novelty of the generated summaries without sacrificing their quality.

(4) *Datasets.* Seq2Seq-based summarization models are widely trained and evaluated on News corpora [44, 50, 98, 100]. However, the journalistic writing style promotes the leading paragraphs of most news articles as summaries [67], which causes the models to favor extraction rather than abstraction [41, 44]. To alleviate this problem, researchers have introduced several datasets from

other domains [28, 63, 67, 121]. In the future, many new datasets will likely be released to build better abstractive summarization systems.

(5) *Evaluation*. Most automatic evaluation protocols, e.g., ROUGE and BERTScore [154], are not sufficient to evaluate the overall quality of generated summaries [70, 91]. We still have to access some critical features, like *factual correctness* [70], *fluency*, and *relevance* [19], of generated summaries by human experts. Thus, a future research direction along this line is building better evaluation systems that go beyond current metrics to capture the most important features which agree with humans. For example, some attempts have been made for generic text generation [120, 140].

REFERENCES

[1] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. 2017. Text summarization techniques: A brief survey. *arXiv preprint arXiv:1707.02268* (2017).

[2] Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086* (2016).

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[4] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. 2016. End-to-end attention-based large vocabulary speech recognition. In *Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 4945–4949.

[5] Lalit Bahl, Peter Brown, Peter De Souza, and Robert Mercer. 1986. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'86)*, Vol. 11. IEEE, 49–52.

[6] Hareesh Bahuleyan, Lili Mou, Olga Vechtomova, and Pascal Poupart. 2018. Variational Attention for Sequence-to-Sequence Models. In *COLING*.

[7] David Balduzzi and Muhammad Ghifary. 2016. Strongly-typed recurrent neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*. JMLR. org, 1292–1300.

[8] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*. 1171–1179.

[9] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research* 3, Feb (2003), 1137–1155.

[10] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* 5, 2 (1994), 157–166.

[11] Adam L, Berger, Vincent J, Della Pietra, and Stephen A, Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics* 22, 1 (1996), 39–71.

[12] Neelima Bhatia and Arunima Jaiswal. 2016. Automatic text summarization and it's methods-a review. In *Proceedings of the 2016 6th International Conference on Cloud System and Big Data Engineering (Confluence)*. IEEE, 65–72.

[13] Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 632–642.

[14] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2016. Quasi-recurrent neural networks. *arXiv preprint arXiv:1611.01576* (2016).

[15] Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Vol. 1. 1662–1675.

[16] Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the CNN/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 2358–2367.

[17] Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling documents. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. AAAI Press, 2754–2760.

[18] Xiuying Chen, Shen Gao, Chongyang Tao, Yan Song, Dongyan Zhao, and Rui Yan. 2018. Iterative document representation learning towards summarization with polishing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 4088–4097.

[19] Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 675–686.

[20] Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 484–494.

[21] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).* 1724–1734.

[22] Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. 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.* 93–98.

[23] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014.*

[24] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems.* 2980–2988.

[25] Tong Lee Chung, Bin Xu, Yongbin Liu, and Chunping Ouyang. 2018. Main point generator: Summarizing with a focus. In *Proceedings of the International Conference on Database Systems for Advanced Applications.* Springer, 924–932.

[26] André Cibils, Claudiu Musat, Andreea Hossman, and Michael Baeriswyl. 2018. Diverse beam search for increased novelty in abstractive summarization. *arXiv preprint arXiv:1802.01457* (2018).

[27] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2019. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations.*

[28] Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. 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)*, Vol. 2. 615–621.

[29] Vipul Dalal and Latesh G. Malik. 2013. A survey of extractive and abstractive text summarization techniques. In *Proceedings of the 2013 6th International Conference on Emerging Trends in Engineering and Technology (ICETET).* IEEE, 109–110.

[30] Dipanjan Das and André F. T. Martins. 2007. A survey on automatic text summarization. *Literature Survey for the Language and Statistics II course at CMU* 4 (2007), 192–195.

[31] Yann N, Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the International Conference on Machine Learning.* 933–941.

[32] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).* 4171–4186.

[33] Carl Doersch. 2016. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908* (2016).

[34] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems.* 13042–13054.

[35] Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science* 14, 2 (1990), 179–211.

[36] Alexander Richard Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.* 1074–1084.

[37] Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation.* 45–54.

[38] Mahak Gambhir and Vishal Gupta. 2017. Recent automatic text summarization techniques: A survey. *Artificial Intelligence Review* 47, 1 (2017), 1–66.

[39] Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. 2017. A convolutional encoder model for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 123–135.

[40] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the International Conference on Machine Learning.* 1243–1252.

[41] Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing.* 4098–4109.

[42] Kevin Gimpel, Dhruv Batra, Chris Dyer, and Gregory Shakhnarovich. 2013. A systematic exploration of diversity in machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing.* 1100–1111.

[43] Alex Graves and Navdeep Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the International Conference on Machine Learning.* 1764–1772.

[44] Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Vol. 1. 708–719.

[45] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. 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)*, Vol. 1. 1631–1640.

[46] Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 140–149.

[47] Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2018. Soft layer-specific multi-task summarization with entailment and question generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 687–697.

[48] Shengbo Guo and Scott Sanner. 2010. Probabilistic latent maximal marginal relevance. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 833–834.

[49] Taher H. Haveliwala. 2002. Topic-sensitive pagerank. In *Proceedings of the 11th International Conference on World Wide Web*. ACM, 517–526.

[50] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. 1693–1701.

[51] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. Volume 1, Number 2.

[52] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[53] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751* (2019).

[54] Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. *arXiv preprint arXiv:1805.06266* (2018).

[55] Dichao Hu. 2018. An introductory survey on attention mechanisms in NLP problems. *arXiv preprint arXiv:1811.05544* (2018).

[56] Luyang Huang, Lingfei Wu, and Lu Wang. 2020. Knowledge graph-augmented abstractive summarization with semantic-driven cloze reward. *arXiv preprint arXiv:2005.01159* (2020).

[57] Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462* (2016).

[58] Daphne Ippolito, Reno Kriz, Joao Sedoc, Maria Kustikova, and Chris Callison-Burch. 2019. Comparison of diverse decoding methods from conditional language models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 3752–3762.

[59] Yichen Jiang and Mohit Bansal. 2018. Closed-book training to improve summarization encoder memory. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 4067–4077.

[60] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099* (2016).

[61] Yaser Keneshloo, Naren Ramakrishnan, and Chandan K. Reddy. 2019. Deep transfer reinforcement learning for text summarization. In *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM, 675–683.

[62] Yaser Keneshloo, Tian Shi, Naren Ramakrishnan, and Chandan K. Reddy. 2018. Deep reinforcement learning on sequence to sequence models. *arXiv preprint arXiv:* (2018).

[63] Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. 2019. Abstractive summarization of reddit posts with multi-level memory networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2519–2531.

[64] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[65] Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[66] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations* (2017), 67–72.

[67] Mahnaz Koupaee and William Yang Wang. 2018. Wikihow: A large scale text summarization dataset. *arXiv preprint arXiv:1810.09305* (2018).

[68] Jonathan Krause, Justin Johnson, Ranjay Krishna, and Li Fei-Fei. 2017. A hierarchical approach for generating descriptive image paragraphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 317–325.

[69] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. 1097–1105.

[70] Wojciech Kryscinski, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Neural text summarization: A critical evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 540–551.

[71] Wojciech Kryściński, Romain Paulus, Caiming Xiong, and Richard Socher. 2018. Improving abstraction in text summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 1808–1817.

[72] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461* (2019).

[73] Chenliang Li, Weiran Xu, Si Li, and Sheng Gao. 2018. Guiding generation for abstractive text summarization based on key information guide network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, Vol. 2. 55–60.

[74] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 110–119.

[75] Jiwei Li and Dan Jurafsky. 2016. Mutual information and diverse decoding improve neural machine translation. *arXiv preprint arXiv:1601.00372* (2016).

[76] Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562* (2016).

[77] Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 1192–1202.

[78] Piji Li, Lidong Bing, and Wai Lam. 2018. Actor-critic based training framework for abstractive summarization. *arXiv preprint arXiv:1803.11070* (2018).

[79] Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep recurrent generative decoder for abstractive text summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2091–2100.

[80] Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. *Text Summarization Branches Out* (2004).

[81] Junyang Lin, Xu Sun, Shuming Ma, and Qi Su. 2018. Global encoding for abstractive summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, 163–169.

[82] Jeffrey Ling and Alexander Rush. 2017. Coarse-to-fine attention models for document summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*. 33–42.

[83] Linqing Liu, Yao Lu, Min Yang, Qiang Qu, Jia Zhu, and Hongyan Li. 2017. Generative adversarial network for abstractive text summarization. *arXiv preprint arXiv:1711.09357* (2017).

[84] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 4487–4496.

[85] Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 3721–3731.

[86] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[87] Elena Lloret and Manuel Palomar. 2012. Text summarisation in progress: A literature review. *Artificial Intelligence Review* 37, 1 (2012), 1–41.

[88] Konstantin Lopyrev. 2015. Generating news headlines with recurrent neural networks. *arXiv preprint arXiv:1512.01712* (2015).

[89] Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 1412–1421.

[90] Inderjeet Mani and Mark T. Maybury. 1999. *Advances in Automatic Text Summarization*. MIT press.

[91] Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. *arXiv preprint arXiv:2005.00661* (2020).

[92] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language Decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730* (2018).

[93] Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 319–328.

[94] Yajie Miao, Mohammad Gowayyed, and Florian Metze. 2015. EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. In *Proceedings of the 2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 167–174.

[95] Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

[96] N. Moratanch and S. Chitrakala. 2016. A survey on abstractive text summarization. In *Proceedings of the 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*. IEEE, 1–7.

[97] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. 3075–3081.

[98] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Ça glar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. *CoNLL 2016* (2016), 280.

[99] Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-Scale Knowledge Extraction*. Association for Computational Linguistics, 95–100.

[100] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 1797–1807.

[101] Preksha Nema, Mitesh M. Khapra, Anirban Laha, and Balaraman Ravindran. 2017. Diversity driven attention model for query-based abstractive summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 1063–1072.

[102] Ani Nenkova and Kathleen McKeown. 2011. *Automatic Summarization*. Now Publishers Inc.

[103] Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics, 160–167.

[104] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank Citation Ranking: Bringing Order to the Web*. Technical Report. Stanford InfoLab.

[105] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 311–318.

[106] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the International Conference on Machine Learning*. 1310–1318.

[107] Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, Vol. 2. 646–653.

[108] Ramakanth Pasunuru, Han Guo, and Mohit Bansal. 2017. Towards improving abstractive summarization via entailment generation. In *Proceedings of the Workshop on New Frontiers in Summarization*. 27–32.

[109] Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304* (2017).

[110] Pavan Kartheek Rachabathuni. 2017. A survey on abstractive summarization techniques. In *Proceedings of the International Conference on Inventive Computing and Informatics (ICICI)*. IEEE, 762–765.

[111] Dragomir R. Radev, Eduard Hovy, and Kathleen McKeown. 2002. Introduction to the special issue on summarization. *Computational Linguistics* 28, 4 (2002), 399–408.

[112] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683* (2019).

[113] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732* (2015).

[114] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1179–1195.

[115] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the International Conference on Machine Learning*. 1278–1286.

[116] Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 379–389.

[117] Horacio Saggion and Thierry Poibeau. 2013. Automatic text summarization: Past, present and future. In *Multi-source, Multilingual Information Extraction and Summarization*. Springer, 3–21.

[118] Baskaran Sankaran, Haitao Mi, Yaser Al-Onaizan, and Abe Ittycheriah. 2016. Temporal attention model for neural machine translation. *CoRR* abs/1608.02927 (2016).

[119] Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. 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)*. Association for Computational Linguistics, 1073–1083.

[120] Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. 2020. BLEURT: Learning robust metrics for text generation. *arXiv preprint arXiv:2004.04696* (2020).

[121] Eva Sharma, Chen Li, and Lu Wang. 2019. BIGPATENT: A large-scale dataset for abstractive and coherent summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2204–2213.

[122] Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 1683–1692.

[123] Shi-Qi Shen, Yan-Kai Lin, Cun-Chao Tu, Yu Zhao, Zhi-Yuan Liu, and Mao-Song Sun. 2017. Recent advances on neural headline generation. *Journal of Computer Science and Technology* 32, 4 (2017), 768–784.

[124] Tian Shi, Ping Wang, and Chandan K. Reddy. 2019. LeafNATS: An open-source toolkit and live demo system for neural abstractive text summarization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. 66–71.

[125] Kaiqiang Song, Lin Zhao, and Fei Liu. 2018. Structure-infused copy mechanisms for abstractive summarization. In *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, 1717–1729.

[126] Ilya Sutskever. 2013. *Training Recurrent Neural Networks*. University of Toronto Toronto, Ontario, Canada.

[127] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence-to-sequence learning with neural networks. In *Advances in Neural Information Processing Systems*. 3104–3112.

[128] Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT press.

[129] Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 1054–1059.

[130] Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 1171–1181.

[131] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 76–85.

[132] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. [n.d.]. WaveNet: A generative model for raw audio. In *Proceedings of the 9th ISCA Speech Synthesis Workshop*. 125–125.

[133] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 6000–6010.

[134] Arun Venkatraman, Martial Hebert, and J. Andrew Bagnell. 2015. Improving multi-step prediction of learned time series models. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*.

[135] Rakesh M. Verma and Daniel Lee. 2017. Extractive summarization: Limits, compression, generalized model and heuristics. *Computación y Sistemas* 21 (2017).

[136] Ashwin K. Vijayakumar, Michael Cogswell, Ramprasath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424* (2016).

[137] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*. 2692–2700.

[138] Li Wang, Junlin Yao, Yunzhe Tao, Li Zhong, Wei Liu, and Qiang Du. 2018. A reinforced topic-aware convolutional sequence-to-sequence model for abstractive text summarization. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. AAAI Press, 4453–4460.

[139] Lex Weaver and Nigel Tao. 2001. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 538–545.

[140] Fei Liu, Yang Gao, Christian M. Meyer, Steffen Eger, Wei Zhao, and Maxime Peyrard. 2019. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Hong Kong, China.

[141] Paul J. Werbos. 1990. Backpropagation through time: What it does and how to do it. *Proc. IEEE* 78, 10 (1990), 1550–1560.

[142] Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*. Springer, 5–32.

[143] Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation* 1, 2 (1989), 270–280.

[144] Yuxiang Wu and Baotian Hu. 2018. Learning to extract coherent summary via deep reinforcement learning. In *32nd AAAI Conference on Artificial Intelligence (AAAI'18)*. 5602.

[145] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* (2016).

[146] Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. In *Advances in Neural Information Processing Systems*. 1782–1792.

[147] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning*. 2048–2057.

[148] Yu Yan, Weizhen Qi, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting future N-gram for sequence-to-sequence pre-training. *arXiv preprint arXiv:2001.04063* (2020).

[149] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R. Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*. 5754–5764.

[150] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1480–1489.

[151] Wojciech Zaremba and Ilya Sutskever. 2015. Reinforcement learning neural turing machines-revised. *arXiv preprint arXiv:1505.00521* (2015).

[152] Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2016. Efficient summarization with read-again and copy mechanism. *arXiv preprint arXiv:1611.03382* (2016).

[153] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2019. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. *arXiv preprint arXiv:1912.08777* (2019).

[154] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with Bert. *arXiv preprint arXiv:1904.09675* (2019).

[155] Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 584–594.

[156] Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. 2018. Neural latent extractive document summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 779–784.

[157] Yuhao Zhang, Daisy Yi Ding, Tianpei Qian, Christopher D. Manning, and Curtis P. Langlotz. 2018. Learning to summarize radiology findings. In *Proceedings of EMNLP 2018* (2018), 204.

[158] Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 654–663.

[159] Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 1095–1104.

Received June 2019; revised June 2020; accepted August 2020