IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 22, 2020, accepted December 3, 2020, date of publication December 7, 2020,
date of current version December 16, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3042886

A Syntax-Augmented and Headline-Aware Neural Text Summarization Method

JINGWEI CHENG, FU ZHANG, AND XUYANG GUO
College of Computer Science and Engineering, Northeastern University, Shenyang 110169, China
Corresponding author: Jingwei Cheng (chengjingwei@mail.neu.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61672139.

ABSTRACT With the advent of the information age, excessive information collection leads to information overload. Automatic text summarization technology has become an effective way to solve information overload. This paper proposes an automatic text summarization model, which extends traditional sequence-to-sequence (Seq2Seq) neural text summarization model by using a syntax-augmented encoder and a headline-aware decoder. The encoder encodes both syntactic structure and word information of a sentence in the sentence embedding. A hierarchical attention mechanism is proposed to pay attentions to syntactic units. The decoder is improved by a headline attention mechanism and a Dual-memory-cell LSTM network to achieve a higher quality of generated summaries. We designed experiments to compare the proposed method with baseline models on the CNN/DM datasets. The experiment results show that the proposed method is superior to abstractive baseline models in terms of the scores on ROUGE evaluation metrics, and achieve a summary generation performance comparable to the extractive baseline method. Though qualitative analysis, the summary quality of the propose method is more readable and less redundant, which agrees well with our intuition.

INDEX TERMS Automatic text summarization, attention mechanism, Seq2Seq, syntactic parsing.

I. INTRODUCTION

Advancements in digital technologies have revolutionized the way information is produced and delivered, and people are confronted with overwhelming information every day, which is far beyond the range that people can efficiently handle and digest. This situation is called *information overload*. Efficiently extracting and understanding valuable information has become an urgent need, which bring into the birth of automatic text summarization technology.

Automatic Text summarization is a process that takes source texts as input and outputs the most important content in a condensed form [1]. People may get the fist of texts in a shorter time by only reading text summaries. In the 1950s, Luhn proposed the first automatic text summarization system [2]. Limited by the computer performance at that time, only statistical information such as word frequency is used to extract sentences in the original text. From then on, automatic text summarization began to attract attention of researches [3].

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

Automatic text summarization methods can be broadly divided into *extractive* methods and *abstractive* methods. Extraction methods directly extract sentences or segments from the original text to form a summary, whereas abstractive methods generate new sentences according to the semantic information of original texts. With the development of neural networks in recent years [4]–[7], the focus on abstractive summarization methods has gradually surpassed extractive methods [8].

Automatic text summarization also benefits from great achievements made in other branches of natural language processing (NLP) [9]. Among others, leaps were made from statistical machine translation to neural machine translation, from bag-of-words models to large pre-trained context-sensitive deep neural language models. While the achievements above have further promoted the development and application of automatic text summarization technology, how to further improve the generation quality of automatic text summarizers remains a hot topic.

The vast majority of abstractive automatic text summarization models based on deep neural networks adopt sequence-to-sequence (Seq2Seq) frameworks [10]. Seq2Seq frameworks consists of an encoder and a decoder.

Compared to traditional neural networks for text summarization, Seq2Seq frameworks go through a same procedure as people are writing summaries, which contains two steps, ''reading comprehension'' and ''writing a summary''. The encoder encodes important information of an input text sequence, which is called ''reading comprehension'' step. Then, the decoder decodes the important information encoded in the output of encoder into a shorter text sequence, which is the step of ''writing a summary''.

In the ''reading comprehension'' step, a recurrent neural network (RNN) usually serves as the encoder. RNNs process the input text sequentially. From a linguistic point of view, however, each text has its syntactic structure and can be seen as a whole of segments with different meanings formed by a tree structure, where each segment has its own unique attribute, such as attributive clause, noun phrase, etc. To the best of our knowledge, there is no relevant research considering the whole syntactic structure of the input text in summary generation.

In news summarization, a special but dominant application scenario in automatic text summarization, the news headline and category are usually provided along with the news content. The headline contains salient information of the news, but among existing applications of the CNN/DM news dataset, there is no work on leveraging the headline information to assist the summary generation.

In a text, some words/phrases convey more important information and are more likely to appear in a summary than their less important counterparts. To identify important words/phrases, attention mechanism [11] is employed in decoder. The attention mechanism, however, may focus too much on some specific words/phrases and generate them in the summary repeatedly, and incurs the *redundancy* problem.

This paper mainly studies abstractive text summarization and proposes several improvements that address critical problems that are not adequately modeled by the basic Seq2Seq framework. The main contributions are as follows.

First, we proposed a syntax-augmented encoder, which leverages a Recursive Neural Network (ReNN) layer to embed complete syntactic structure information of a sentence and incorporates the structure embedding in the sentence embedding. We also propose a syntactic unit attention mechanism that cooperates with the decoder at the decoding stage to guide summary generation.

Second, for an input text with its headline available, e.g. a news article, we propose headline-aware summary generation method which allows to obtain additional and precise key information from the headline.

Finally, to prevent repeatedly generating same words/phrases, we proposed a Dual-memory-cell LSTM network to record in the decoding stage both the already generated summary and the content to be generated.

The rest of the paper is organized as follows. We describe related work in Section II and explain the proposed method in Section III. We conduct experiments in Section IV and then add summary and discussion in Section V.

II. RELATED WORK

We survey here abstractive text summarization methods using Seq2Seq frameworks, which are most relevant to our work.

In 2015, Rush *et al.* [12] first proposed to use a Seq2Seq neural networks for generating text summaries, which created an era of abstractive text summarization. From then on, the vast majority of neural abstractive summarization models adopt the Seq2Seq framework. In such models, an encoder encodes the input text as a vector representation, which is then fed into a decoder to obtain the summary. Encoder determines how much relevant information is extracted, and decoder determines how much the model can restore the most important information. Therefore, the choice of encoder and decoder directly determines the pros and cons of the text summarization methods.

The first Seq2Seq text summarizer [12] employ a CNN as the encoder and a feed-forward neural network as the decoder. Subsequently, RNN was considered to be a more suitable encoder choice because of its better ability to process sequence input [13], [14]. To further enhance the model's ability to encode long sequences, RNN are replaced with LSTM [15]–[19]. Cao *et al.* [20] used syntactic parsing results to construct a binary tree, and encoded it with a ReNN to obtain a vector representation for sentence extraction. This method is similar to our proposed syntax-augmented encoder, but only uses the traversal order of the binary tree to represent the original word order, which lacks the power of retaining the complete structural information of sentences. In addition, no attention mechanism is used to assist the summary generation. Song *et al.* proposed a LSTM variant [21] and applied it to the generation model from AMR graph to text, as traditional LSTM model will lose the structure information when linearizing the AMR graph. The LSTM variant directly encode the graph-level semantics. Our proposed syntactic parsing tree encoder is partially inspired by this idea.

The decoder also undergoes a process from feedforward neural network, to deep feedforward neural network RNN, and then to LSTM, with the depth of models gradually deeper and the model structure more complicated. Recently, Zhao *et al.* [22] propose a variational neural decoder (VND) text summarization model. The model introduces a series of implicit variables by combining variational RNN and variational autoencoder, which is used to capture complex semantic representation at each step of decoding. The model can better capture the complex semantics and the strong dependence between the adjacent time steps when outputting the summary, thereby improving the performance of generating the summary.

There also have been efforts to improve the performance of Seq2Seq text summarizers. Nallapati *et al.* [13] replicated the success of the attention mechanism in machine translation [11], applied the Seq2Seq model and Soft Attention mechanism to automatic text summarization, and achieved remarkable performance improvements. To cope with the problem of insufficient memory capacity of long text

summaries in traditional RNN encoders, Celikililmaz proposed a multi-agent method [23], where each agent is equivalent to an independent encoder. The original text is divided into multiple parts, each of which uses an agent for processing and multiple agents will pass information in deep layers. In this way, the length of the sequence that RNN need to process decreases, while the full text information is retained as much as possible. This method uses a hierarchical attention mechanism in the decoder to calculate attention weights for different encoders and different words in text sequences processed by each encoder. Existing Seq2Seq models tend to memorize words and patterns in training data sets, rather than learning the meaning of words. Therefore, the generated sentences are usually grammatically correct, but semantically inappropriate. Ma *et al.* [24] introduced a novel Seq2Seq model called word embedding attention network (WEAN). The model generates words by querying the distributed word representations (that is, neural word embeddings), i.e., using a method similar to attention mechanism to select semantically similar words in the vocabulary as outputs. Li *et al.* [25] propose a multi-head attention summarization (MHAS) model to address the problems of duplicate and missing original information. The MHAS model consider the previously predicted words when generating new words to avoid generating a summary of redundant repetition words. And it can learn the internal structure of the article by adding self-attention layer to the traditional encoder and decoder and make the model better preserve the original information. They also integrate the multi-head attention distribution into pointer network creatively to improve the performance of the model. Song *et al.* [26] propose an LSTM-CNN based Abstractive Text Summarization framework (ATSDL) that generate new sentences by extracting and assembling semantic phrases from the original text. They use sequential information of phrases to alleviate the problem of rare words, whereas the structural information of semantic units is utilized in our method. Guo *et al.* [27] propose an MS-Pointer Network, which employs the multi-head self-attention mechanism in the encoder to extract more semantic features for the summary and a pointer network to solve the out of vocabulary problem. Kouris *et al.* [28] propose a novel framework for enhancing abstractive text summarization by combining a traditional seq2seq model with semantic data transformations. The framework consists of three parts, a theoretical model for producing semantic-based generalized summary, and a methodology to transform the generalized summary into human-readable form. The innovation of our method is the improvement of the Seq2Seq framework, i.e., an encoder that combines complete sentence grammatical structure information, and a decoder with an attention on the headline of a input text and a Dual-memory-cell LSTM network that memorize both the already generated and to be generated parts of a summary.

Readers may refer to [9], [29], [30] for a comprehensive discussion on abstractive text summarization.

III. SYNTAX-AUGMENTED AND HEADLINE-AWARE TEXT SUMMARIZATION MODEL

In this section, we introduce an abstractive Seq2Seq summarization model, which consists of a syntax-augmented encoder and a headline-aware decoder. We first introduce the architecture of the model, then the model structure of the encoder and decoder.

A. MODEL ARCHITECTURE

The proposed summarization model is shown in Fig. 1, which consists of an encoder and a decoder:

- In the encoder, for each sentence in an input text, an embedding layer embeds both the sentence and the syntactic parsing tree of the sentence itself into a sentence embedding. All sentence embeddings are then passed to a Bi-LSTM layer to produce an embedding of the input text. In the meantime, a syntactic unit attention mechanism is applied to compute attentions for each syntactic unit in a syntactic parsing tree.
- In the decoder, we use an LDA model to encode the headline of an input text into a headline vector, which is used to calculate a joint attention with the syntactic unit attention from the encoder part. Meanwhile, a Dual-memory-cell LSTM network is used to alleviate the redundancy problem while generating the summary.

We detail these two parts in the following.

B. ENCODER

The encoder consists of an embedding layer for both the sentences and the syntactic parsing trees of sentences, a BiLSTM layer for text encoding, and an attention layer for syntactic units.

1) EMBEDDING LAYER

Suppose we have a set of input texts $T = \langle t_1, \ldots, t_m \rangle$, where t_j denotes the j-th text in T, and $m = |T|$ the number of texts in T. Given a text $t = \langle sent_1, \ldots, sent_n \rangle$ where $sent_i$ is the i-th sentence of text t, n represents the number of sentences in t.

Syntactic parsing is one of the key underlying technologies in natural language processing and its basic task is to determine the syntactic structure of sentences or the dependency between words in sentences. Syntactic parsing tree is one of the representations of the syntactic parsing result of a sentence, which carries more semantic and syntactic information compared with characters, words, or phrases. The syntactic parsing tree of t is denoted as $pt = \langle tree_1, \ldots, tree_n \rangle$, where $tree_i$ denotes the syntactic parsing tree of $sent_i$. We use Stanford CoreNLP [31] to obtain the syntactic parsing tree of a sentence. Fig. 2 shows the syntactic parsing tree of a sentence "I love cats". In syntax, words, phrases and sentences are called *syntactic units*, where the word is the smallest one, and the sentence is the largest one.

In a syntactic parsing tree, each node has a data structure as is shown in Table 1, which contains a syntactic tag, a pointer

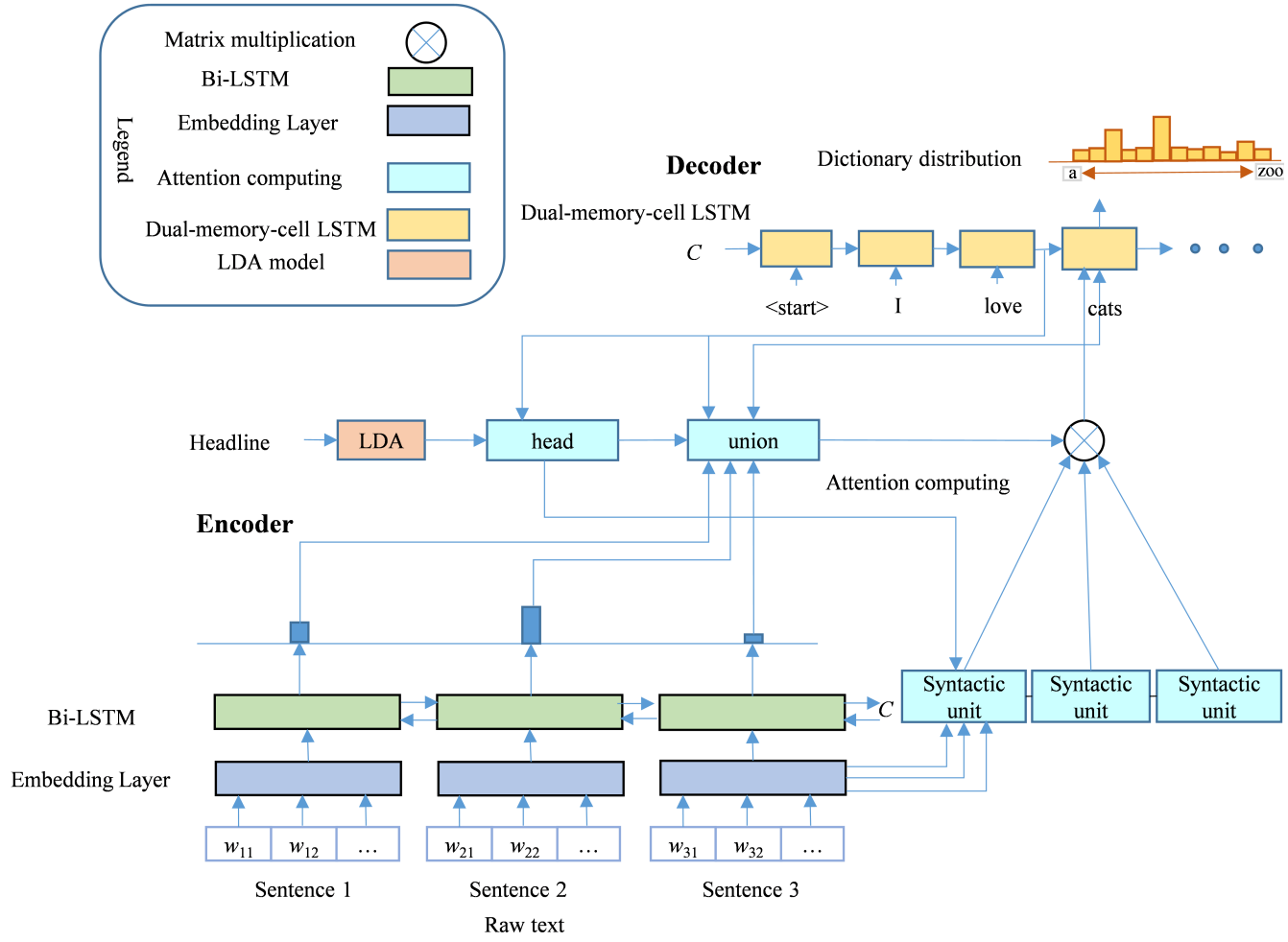


FIGURE 1. Architecture of syntax-augmented and headline-aware text summarization model.

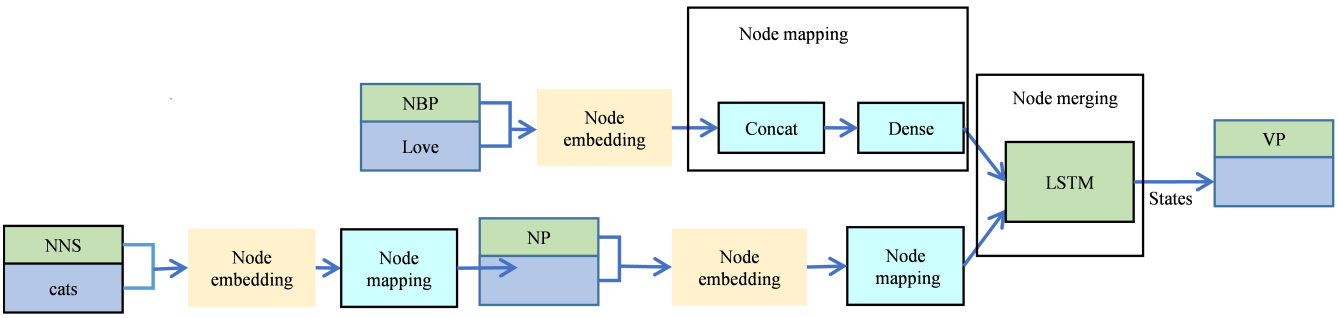


FIGURE 2. An example of a syntactic parsing tree of a sentence.

to its parent node, a word in the sentence for a leaf node, and a reference to each of its child node for a non-leaf node. In Fig. 2, *ROOT*, *S*, *NP* and so on are syntactic tags.

Based on the analysis above, the representation of a text should also retain its syntactic structure information besides the representation of words in the text.

We employ embedding method to avoid the data dimension being too high and to preserve the semantic distribution of the original data. We define three operations to fulfil this task, as is shown in Fig. 3. A *node embedding* operation embed the syntactic unit and the tag of a node into vectors, a *node mapping* operation is used to combine the syntactic unit embedding and the tag embedding, and a *node merging* operation outputs the syntactic unit embedding of a given node by accepting as input the results of node mapping of node embeddings of its child nodes. By recursively using the three operations from the ROOT node of a syntactic parsing tree, we finally acquire the node embedding of ROOT, which is also the embedding of a sentence.

TABLE 1. Data structures of nodes in a syntactic parsing tree.

Class TNode
Attribute:
tag: syntactic tag
children: references to child nodes
word: a word stored in a leaf node
parent: pointer to its parent

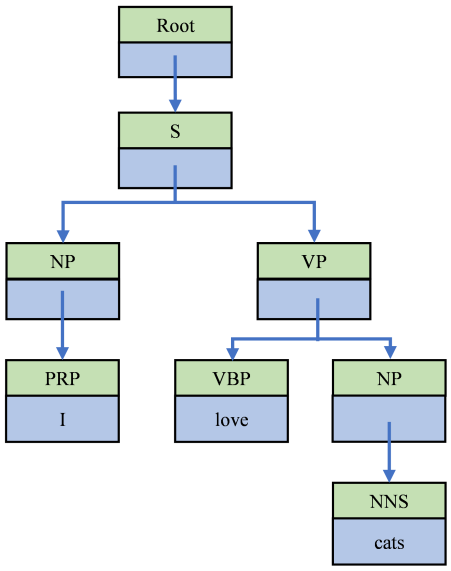


FIGURE 3. An example of a node operations in the embedding layer.

a: NODE EMBEDDING

In Fig. 2, each node contains a syntactic tag to convey its syntactic characteristics, and a syntactic parsing tree implies the syntactic structure of a sentence. For example, a *SINV* tag means that current node is an inverted sentence, a *VP* tag means that current node is a verb phrase, and a *S* tag indicated a sentence. As the number of tags in syntactic parsing trees is much smaller than the number of words and they are semantically different, we use two embedding matrices to embed words and tags respectively.

Assume that $T_{vt\times dt}$ is the embedding matrix for tags, vt is the number of tags, and dt is the dimension of a tag embedding. Assume that $E_{ve\times de}$ is the embedding matrix for words, and ve is the number of words, de is the dimension of a word embedding. The tag embedding and word embedding of leaf node i in a syntactic parsing tree are respectively v_{tag_i} and v_{word_i}:

$$v_{tag_i} = T[tag_i] \tag{1}$$

$$v_{word_i} = E[word_i] \tag{2}$$

Using the above equations, tags and words of leaf nodes in the tree are embedded into vectors. Since embeddings of syntactic units in non-leaf nodes are calculated by the node merging operation (as we will see afterwards) on their child nodes, only embeddings of tags in none-leaf nodes are calculated using (1). To compare with baseline methods in Section IV, instead of using pre-trained word embeddings, we jointly train the embedding layer with the rest layers in our model.

b: NODE MAPPING

In order to retain syntactic structure information for summarization, we combine each syntactic unit with its corresponding tag with the node mapping operation. The syntactic unit embedding and the tag embedding are used as input, and the output is transferred to the node merging operation as input. The output of the ROOT node is used as the representation of the syntactic parsing tree of the sentence. The calculation equation is as follows.

$$v_{node} = \tanh(Dense([v_{tag}; v_{su}])) \tag{3}$$

where [;] denotes concatenation of vectors, v_{su} is the syntactic unit embedding of current node, and v_{node} is the result of the node mapping operation and we call it the *node representation* or *node vector*. We use a tanh activation function to fit the non-linear relation between the syntactic unit embedding and the tag embedding.

There is still a problem to be solved. The syntactic parsing tree for a complex sentence may have a large depth. When using backpropagation to train parameters of this model, too deep the tree structure makes it hard for the loss to propagate to the deeper part of the parsing tree. We thus add the average value of node representations of its children, as is shown below.

$$v_m = v_m + sum([v_1, \ldots, v_n]) \tag{4}$$

where v_i represents the node representation of the i-th child of current node.

c: NODE MERGING

For a node in a syntactic parsing tree, the syntactic unit in it is composed of syntactic units in its child nodes, down to the words of leaf nodes. To get the embedding of a syntactic unit, we need to process each syntactic unit composing it in sequence, and thus use a LSTM layer which is more suitable for processing such sequence information. As the node merging operation is same for all syntactic units, a same LSTM is used with shared weights.

Suppose that a node in a parsing tree contains a sequence of child nodes with $\langle v_1, \ldots, v_n\rangle$ their node representations, then its syntactic unit embedding is calculated as

$$h_{out} = LSTM(\langle v_1, \ldots, v_n\rangle, h_0) \tag{5}$$

where the initial state h_0 of the LSTM network is set to zero.

A sentence embedding thus can be obtained by recursively performing the above three node operations on the parsing tree of the sentences in the input text, starting from the root node. Syntactic parsing trees cannot be handled by ordinary neural networks. We therefore use ReNN to map the syntactic parsing tree of a sentence to a sentence embedding.

The node representations and sentence embedding are passed to the decoder for calculating the attention weights in the decoding stage.

2) BiLSTM LAYER

After obtaining embeddings of sentences in a text t, we use a BiLSTM (Bidirectional Long Short Term Memory) layer to calculate the semantic representation of the text t. BiLSTM is a variant of LSTM that has been shown to perform well on various sequential input tasks. BiLSTM maintains two separate states that are generated by two different LSTMs by feeding in inputs forwardly and backwardly. The idea behind bi-directional network is to capture information of contextual information. In comparison to LSTM, BLSTM has two networks and run inputs in two ways, one from past to future and another from future to past, whereas LSTM has one network which processes inputs forwardly only.

3) ALGORITHM OF ENCODER

The algorithm of syntax-augmented encoder is shown in Algorithm 1.

Algorithm 1 Syntax-Augmented Encoder Algorithm

Input: Syntactic parsing trees $pt = \langle tree_1, \ldots, tree_n \rangle$ of the input text $t = \langle sent_1, \ldots, sent_n \rangle$, each with a root node $root_i$
Output: vector of doc, hidden of sentence

1 **for** *each* $tree_i \in t$ **do**
2 $node\_vecs_i = \emptyset$;
3 $sent\_vec_i, node\_vecs_i = GetVec(root_i)$;
4 **end**
5 $t\_vec = Bi\text{-}LSTM(\{sent\_vec_i\})$;
6 **return** $t\_vec, \{node\_vecs_i\}$;

In line (2), we define, for each syntactic parsing tree $tree_i$, an empty vector list $nodes\_vec_i$ for accommodating vectors of nodes in $tree_i$. Then, in line (3), by using a recursive function *GetVec* (see Algorithm 2), we obtain the representation of $tree_i$ as well as node vectors. In line 5, we use BiLSTM on all sentence vectors to obtain the representation t_vec of the text t. The obtained node vectors and the text vector are passed to the decoder as parameters in attention weight calculation.

Line (3) and (4) are the process of Node Merging, line (7) corresponds Node Embedding, Whereas line (9) is Node Mapping operation.

4) HIERARCHICAL ATTENTION MECHANISM

We have incorporated the structural information of the input text in the encoder. In fact, this structural information can also be used in the decoder. We propose a syntactic unit attention mechanism, which provides attention on nodes in a syntactic parse tree and is beneficial to obtain richer semantic information in the decoding stage. It is different from traditional attention mechanism that only provide attention on words [11], [32].

In a typical syntactic parsing tree, most non-leaf nodes are phrases, whereas leaf nodes are individual words. In order to obtain phrase information in decoding stage, node representations are used for attention calculation in the decoding stage.

Algorithm 2 GetVec Function

Input: A node *node* of a syntactic parsing tree
Output: The representation of the node and its subnodes

1 $sub\_nodes = \emptyset$;
2 **if** *node has children* **then**
3 $sub\_nodes = [GetVec(child)$ for *child* in $node.children]$;
4 $sub\_node\_vecs = LSTM(sub\_nodes)$;
5 **end**
6 **else**
7 $sub\_node\_vecs = Embedding(node.word)$;
8 **end**
9 $node\_vec = MLP(sub\_node\_vecs) + sum(sub\_nodes)/|node.children|$;
10 $node\_vecs.append(node\_vec)$;
11 **return** $node\_vecs$;

Considering texts in CNN/DM dataset are all news articles, which are of a longer length and contain more sentences than regular texts, we employ a hierarchical attention mechanism. The first layer calculates attention weights on sentences, and the second layer calculates attention weights on syntactic units in each sentence. For example, for a common RNN decoder, the attention calculation for the time step in the decoder is as follows.

$$e_{ij} = v_1^T \tanh\left(W_1 s_{i-1} + W_2 node_j\right) \tag{6}$$

$$sent_{ij} = v_1^T \tanh\left(W_3 s_{i-1} + W_4 output_j\right) \tag{7}$$

$$node\_wt_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{N_i} \exp\left(e_{ik}\right)} \tag{8}$$

$$sent\_wt_i = \frac{\exp\left(sent_{tj}\right)}{\sum_{k=1}^{n} \exp\left(sent_{tk}\right)} \tag{9}$$

$$sent\_ctx_i = \sum_{j=1}^{N_i} node\_wt_{ij} \times node_{ij} \tag{10}$$

$$ctx\_vec_t = \sum_{i=1}^{n} sent\_wt_i \times sent\_ctx_i \tag{11}$$

where s_{i-1} is the output of the $i-1$ time step of the decoder, $node_{ij}$ represents the j-th node in the syntactic parsing tree of the i-th sentence, and $output_j$ represents the output state of the j-th sentence. Finally, the output of the next time step is predicted by using the syntactic unit attention mechanism in the decoder.

$$s_t = Decoder(s_{t-1}, context\_vector_t) \tag{12}$$

After calculating the attention weight on each sentence and the attention weight on each syntactic unit within each sentence, the two are weighted and summed to get the attention vector of syntactic units related to the original text.

C. HEADLINE-AWARE DECODER

In automatic text summarization, how to determine the important information in the input text is an important issue. In the field of news summarization, where automatic text summarization technologies are widely used, a news article often has a manually-written headline, which generally contains the central idea of the text. We thus propose a headline-aware decoder that uses headline information during the decoding stage to generate a summary that is closer to the important information of the input text.

When generating text summaries, as important information is assigned greater attention weights, the generated summary always has repetitive content. The use of headline-aware method will aggravate this situation. We propose a Dual-memory-cell LSTM to alleviate the redundancy in summaries. The decoder part in Fig. 1 shows the headline-aware method and the improved LSTM.

1) HEADLINE EMBEDDINGS

We first use an LDA topic model [33] to transform the headline into a topic vector, through which we can generate summary content closer to the topic in the decoding stage. LDA is a generative probability model of a corpus. It assumes each text **w** in a corpus is generated by the following process:

1) Choose $N \sim \text{Poisson}(\xi)$, i.e., choose the length N of **w** from a Poisson distribution with parameter ξ.
2) Choose $\theta \sim \text{Dir}(\alpha)$, i.e., choose a parameter θ from a Dirichlet distribution with parameter α.
3) For each word w_n in **w**, a
   a) Choose a topic $z_n \sim \text{Multinomial}(\theta)$, i.e., choose a topic from a multinomial distribution with parameter θ.
   b) Choose a word w_n from $p(w_n|z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

LDA topic model assumes the relationship between each text and multiple topics is a multinomial distribution, which is denoted as Multinomial(θ). The relationship between each topic and the thesaurus is also a multinomial distribution conditioned on the topic, which is denoted as $p(w_n|z_n, \beta)$. By training these two distributions on headline-topic pairs in the training data, we obtain the topic distribution vectors for all headlines.

2) HEADLINE-AWARE ATTENTION MECHANISM

The topic distribution vector of a headline is then added to the summarization model. In the decoding stage, the decoder calculate attention not only on the output of previous step, but also on the headline at current step, and the concatenation of the two attention results serve as part of the input of next step.

Just as usually happens in regular attention mechanism, adding attention on headlines may lead to repetitive generation of headline content in the summary. To avoid this, we need to leverage already generated information. Therefore, different from the regular attention mechanism, we use hidden states of previous time step of the decoder when calculating attention on the headline as follows.

$$v_h = LDA(headline\_text) \tag{13}$$

$$d = v_a^T \tanh\left(W_1 v_h + W_2 s^{t-1}\right) \tag{14}$$

$$\beta_j = \frac{\exp\left(ds_j^{t-1}\right)}{\sum_{k=1}^{m} \exp\left(ds_k^{t-1}\right)} \tag{15}$$

$$c_t = \sum_{j=1}^{m} \beta_j s^{t-1} \tag{16}$$

where m represents the size of decoder memory cell, s^{t-1} represents the memory information output of previous step. This attention mechanism calculates the headline's attention on the memory information. We combining the headline attention with regular attention as follows.

$$s_t' = Attention\left([x_t; s_t; c_t], encoder\_states\right) \tag{17}$$

$$h_t = Decoder\left(x_t, s_t'\right) \tag{18}$$

3) DUAL-MEMORY-CELL LSTM

When applying Seq2Seq framework in generation tasks, such as text summarization and machine translation, we usually use LSTM as decoder with attention mechanism to obtain the weighted averaged attention of the output of encoder. However, this kind of network is prone to generate duplicate text, which makes the generated summary highly redundant. In the headline-aware decoder, the headline contains highly central information, which will aggravate duplicate output of salient contents in the original text. We propose a Dual-memory-cell LSTM to alleviate the problem of output redundancy. The network structure of it is shown in Fig. 4.

When people write summaries, they often first read for understanding the article, and then write the summary based on their understanding. During the writing process, they compare the full text with the content they write to determine the content to write subsequently. However, traditional LSTM network cannot simultaneously memorize the output of the previous time step and global information.

In a text summarization decoder using traditional LSTM, the input of the memory cell in the first time step is from the last state of the memory cell of the encoder, which can be regarded as containing full semantic information of the original text. With the decoding proceeding, the memory cell of each time step contains the residual information of the original text (to be generated). However, the decoder is unable to associate the already generated information with information to be generated.

We thus propose a Dual-memory-cell LSTM network. Compared with the traditional LSTM network, the input gate, forget gate and output gate of the improved LSTM network remain unchanged. The forget, input and output vectors are calculated by using the hidden state output of previous time

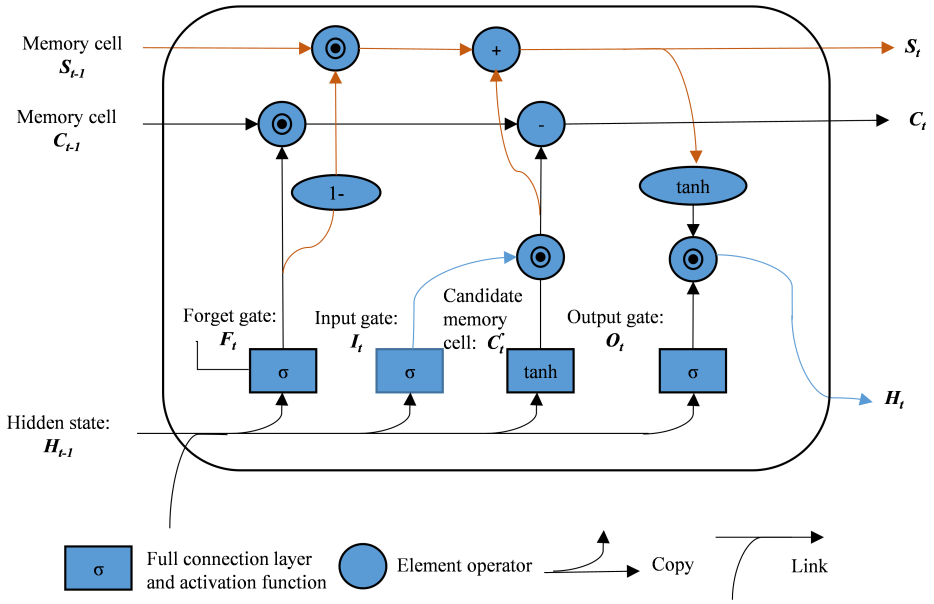


FIGURE 4. Dual-memory-cell LSTM.

step and the input of current time step as follows.

$$F_t = \sigma\left(W_f X_t + U_f H_{t-1} + b_f\right) \tag{19}$$

$$I_t = \sigma\left(W_i X_t + U_i H_{t-1} + b_i\right) \tag{20}$$

$$O_t = \sigma\left(W_o X_t + U_o H_{t-1} + b_o\right) \tag{21}$$

In the network (see Fig. 4), the memory cell C memorizes the content to be generated. The state update of memory cell C remains the same as regular forget method, whereas its input changes from the original addition by elements to the subtraction by elements.

$$C_t = F_t \circ C_{t-1} - I_t \circ \tanh\left(W_c X_t + U_c H_{t-1} + b_c\right) \tag{22}$$

The newly introduced memory cell S is used to memorize the already generated content. In contrast to memory cell C, in the forget gate, the content forgotten by C should be retained. Therefore, we use 1 subtract by elements the forget gate output F_t to obtain the forgotten content opposite to C. In addition, to memorize all the output, it is necessary to keep in S the subtracted content from C.

$$S_t = (1 - F_t) \circ S_{t-1} + I_t \circ \tanh\left(W_c X_t + U_c H_{t-1} + b_c\right) \tag{23}$$

Combining the memories of two memory cells, the output of current time step is calculated as follows.

$$H_t = O_t \circ \tanh\left([C_t : S_t]\right) \tag{24}$$

The Dual-memory-cell LSTM network use two memory cells to respectively transmit and process the information to be generated and already generated in a certain time step. By avoiding generating redundant information, it generates other important information. Furthermore, combining with the headline-aware attention, more accurate attention calculation results for the full-text information are obtained.

TABLE 2. CNN/DM dataset.

Dataset	CNN	Daily Mail	Total
training set	83,568	196,557	280,125
valid set	1,220	12,147	13,367
test set	1,093	10,396	11,489
number of sentences per text	29.8	26.0	27.1
number of sentences per summary	3.54	3.84	3.75
number of words per text	732.7	747.2	742.9
number of words per summary	46.68	55.43	52.8

IV. EXPERIMENTS

We introduce our setups of the CNN/DM dataset, present the baseline methods, and finally analyze experimental results. The model is implemented with MXNet [34] and the source code is available at: https://github.com/theDoctor2013/SA-HA-Sum.

A. DATASET AND EXPERIMENT SETUP

1) DATASETS

The CNN/DM dataset [35] is one of most common datasets for abstractive text summarization task, which collects 100000 news data from CNN website and about 200000 news data from Daily Mail website. The dataset scale and division quantity are shown in Table 2.

2) MODEL SETUPS

We adopt a general *sequence loss* function which is the average of the vocabulary losses in the actual summary. The *vocabulary loss* of i-th word is the cross-entropy loss over

TABLE 3. Hyperparameters of syntax-augmented encoder.

Parameter	Value	Parameter	Value
vocab_size	20k	word_emb_size	256
merging_LSTM_dim	128	tag_emb_size	128
epoch_num	5	dense_layer_size	128
BiLSTM_dim	256	sent_attn_units	128
su_attention_units	128	batch_size	1
beam_size	64	dropout	0.5
optimizer	adam	learning_rate	0.002

TABLE 4. Hyperparameters of headline-aware decoder.

Parameter	Value	Parameter	Value
vocab_size	20k	word_emb_size	256
topic_num	64	decoder_lstm	256
headline_attn_units	128	regular_attn_units	128
batch_size	16	beam_size	64
dropout	0.5	learning_rate	0.001
epoch_num	10	lda_iter	50
optimizer	adam		

vocabulary.

$$p_i = Softmax\left(pred_i\right) \tag{25}$$

$$loss = -\frac{1}{L}\sum_{i=1}^{L} label_i \log\left(p_i\right) \tag{26}$$

where $pred_i$ is the i-th step output of text summarization model, and $label_i$ is the embedding of i-th word in the reference summary, L is the length of the reference summary.

The hyperparameters used in the model are shown in the following two tables, where Table 3 contains hyperparameters for the syntax-augmented encoder, and Table 4 contains hyperparameters for the headline-aware decoder. In our experiments, we choose not to use pre-trained word embeddings, but use the *word_emb_size* hyperparameter in Table 3 (Table 4) to construct a embedding layer in the encoder (decoder) and train the parameters jointly with other layers. The hyperparameters are self-explained by their names. For example, *su_attn_units* stands for syntactic unit attention unit size.

As the structure of the syntactic parsing tree of each sentence varies, it is impossible to process multiple sentences at the same time. We thus specify *batch_size* = 1 when training the Syntax-augmented encoder. The *batch_size* for Headline-aware decoder is specified according to model complexity and the memory size of GPU. The *epoch_num* is determined when the validation log-likelihhood does not improve for an epoch. We choose optimal *learning_rate* with the Adam initial learning rate in {0.1, 0.07, 0.02, 0.01, 0.005, 0.002, 0.001}. We choose optimal *BiLSTM_dim* and *decoder_lstm* from {128, 256, 512} and optimal *merging_LSTM_dim*, *sent_attn_units* and *su_atten_units* from {64, 128, 256}.

Other hyperparameters are determined based on experience. For example, the *vocab_size* is set to 20k, which covers most commonly used words; *topic_num* is set to 64 based on the number of categories in most news datasets. The beam size is set to 64 by considering the balance between performance and training time.

3) BASELINE METHODS

In order to verify the validity of proposed model, we compare with three classic baseline methods.

- A summarization model proposed by Rush *et al.* [12] with an attention-based encoder.
- The Seq2Seq summarization model proposed by Nallapati *et al.* [13], where the encoder uses a bidirectional GRU and the decoder performs attention calculation on all hidden states of the encoder. The reason we select this model for comparison is that the Seq2Seq framework used in this model has no additional structure, and can be compared with the proposed model.
- Lead-3 [36] is an extractive method which directly extract the first three sentences of the input text as the summary. Using this method as a baseline is to compare models with the simplest extraction method to the performance of our abstractive summarization method.

4) EVALUATION METRICS

We use one of the most popular text summary evaluation methods ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [37]. This method discriminates the quality of computer-generated summaries by comparing word sequences that occur simultaneously in a computer-generated summary and human-written reference summaries. It contains multiple evaluation strategies, ROUGE-N, ROUGE-L, ROUGE-W, and ROUGE-S. We use ROUGE-N and ROUGE-L in our experiments.

ROUGE-N is computed as follows:

$$ROUGE\text{-}N = \frac{\sum_{S\in RS}\sum_{gram_n\in S} Count_{match}(gram_n)}{\sum_{S\in RS}\sum_{gram_n\in S} Count(gram_n)} \tag{27}$$

where the denominator is the sum of the number of N-grams occurring in the reference summaries, and the numerator is the number of N-grams shared by a computer-generated summary and reference summaries. Commonly used ROUGE-N are ROUGE-1 and ROUGE-2. ROUGE-L is a metrics of Longest Common Subsequence (LCS). ROUGE-L is calculated as follows:

$$R_{lcs} = \frac{LCS(X,Y)}{m} \tag{28}$$

$$P_{lcs} = \frac{LCS(X,Y)}{n} \tag{29}$$

$$F_{lcs} = \frac{\left(1+\beta^2\right)R_{lcs}P_{lcs}}{R_{lcs}+\beta^2 P_{lcs}} \tag{30}$$

where $LCS(X, Y)$ is the length of a longest common subsequence of X and Y, and m and n are the length of X and Y.

B. EXPERIMENT RESULT ANALYSIS

We conduct four different types of experiments.

First, we conduct an ablation experiment to show the effect of different modules in our model, i.e., SA which indicates the model uses a Syntax-augmented encoder, HA which stands for the Headline-Aware attention, and DLSTM which indicates the Dual-memory-cell LSTM is used. HA-DLSTM, SA-DLSTM and SA-HA-LSTMS are combinations of two or three above modules. The experimental results on CNN/DM dataset are shown in Table 5, which are F1 values. The upper three rows are results of baseline models.

TABLE 5. Comparison of rouge scores on CNN/DM dataset.

Model Name	ROUGE-1	ROUGE-2	ROUGE-L
Rush [12]	29.78	11.89	26.97
Nallapati [13]	35.46	13.30	32.65
Lead-3 [36]	**39.2**	**15.7**	35.5
SA	36.64	13.9	34.05
HA-DLSTM	37.83	14.17	33.68
SA-DLSTM	38.25	13.57	34.67
SA-HA-DLSTM	38.94	15.38	**35.69**

We can see from Table 5 that, by introducing Syntax-augmented encoder (SA), our model achieves much better ROUGE scores than abstractive baseline models [12], [13]. Our model with SA and DLSTM improves the ROUGE scores further. Our best model (SA-HA-DLSTM) surpasses baseline models more than 3 points in average.

As is shown in Table 5, Our best model (SA-HA-DLSTM) achieves higher score than the extractive baseline Lead-3 on ROUGE-L metrics and achieves comparable scores on ROUGE-1 and ROUGE-2 scores. Maybe it is because abstractive models predict the output words according to the word vector distribution, though similar to in semantics, but fails to generate the original words. Another possible reason is not dealing with OOV words. Therefore, compared with the Lead-3 method, the short segments (1-gram or 2-gram) generated by our models are often different from those in the reference summaries, which may affect the generation accuracy of the segments evaluated by ROUGE-1 and ROUGE-2 to a certain extent. In addition, due to the writing habits of news articles, which often summarize the full text at the beginning, Lead-3 achieves higher scores on CNN/DM dataset.

To the best of our knowledge, most up-to-date abstractive methods fail to outperform extractive methods on CNN/DM dataset [26], [27]. In addition, See *et al.* [15] also offered two explanations for these observations. Firstly, news articles tend to place important information at the beginning, which partially explains the strength of the lead-3 baseline. Secondly, the nature of the task and the ROUGE metric make extractive approaches and the lead-3 baseline difficult to beat.

The second experiment compares the summary redundancy generated by traditional LSTM and our proposed Dual-memory-cell LSTM. The redundant text problem has always been a hard-to-crack issue in neural-network-based text summarization model, that is, there are repetitive content in the generated summary. This is because the attention weight and generation operation in complex networks always incline to focus only on important information in the original text, which lead to repetitively generating same important contents.

We compare the proportions of the duplicate segments in the output summaries of traditional LSTM decoder and our proposed Dual-memory-cell LSTM as decoder, and the reference summaries, as is shown in Fig. 5.

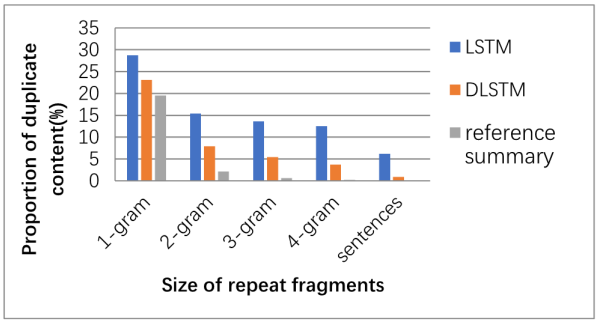


FIGURE 5. Comparison of summary redundancy.

We compare repetitions of five segment lengths, namely, 1-gram, 2-gram, 3-gram, 4-gram and whole sentence repetition. The model using traditional LSTM decoder has the highest repetition rate. When using Dual-memory-cell LSTM, the repetition rates of all segment lengths drop significantly, which proves its effectiveness. The Dual-memory-cell LSTM, by recording the output history of the decoder with a newly introduced memory cell, retains the history information of already generated summary and deletes the information from memory cell of to be generated.

The third experiment is a qualitative analysis of generated summaries. We select several news articles from CNN test set to evaluate if the proposed methods have achieved improvement in the quality of summaries. We compared the reference summary with summaries generated by a baseline method and our proposed methods. One of the results is shown in Table 6. The news describes a total eclipse that occurred at 4:58 a.m. pacific time on the news day. However, in the baseline model, the generated summary fails to generate the exact time the total lunar eclipse occurred. In our proposed model, the time of the total lunar eclipse is correctly generated. At the same time, since the ''shortest total lunar eclipse'' is mentioned in the headline, the summary generated by the headline-aware decoder extracts descriptions about ''the shortest total lunar eclipse of the century''. In addition, from Table 6, in the summary generated by the baseline model, the sentence ''a total lunar eclipse for nearly five minutes.'' is repeatedly generated, whereas there is no serious redundancy in our methods.

The fourth experiment is the human evaluation of the readability, relevance, and grammatical correctness. We invite 5 students in our lab to serve as human evaluators. To perform this evaluation, we randomly sampled 100 examples from the CNN/DM dataset, each contains the original article, the reference summary as well as summaries generated by

TABLE 6. Comparison of summaries by different models.

Headline: Stargazers enjoy shortest total lunar eclipse of the century
Excerpt from the original text: (cnn) the third blood moon in a four-Section series was the shortest eclipse of the bunch , but still a sweet treat for early risers in north america . the moon slipped fully into earth 's shadow at 4:58 a.m. pacific time (7:58 a.m . et) saturday , starting a total lunar eclipse for nearly five minutes – what nasa says will be the shortest such eclipse of the century . the celestial body took on a burnt-orange tint in the minutes before , during and after the total eclipse , giving the moon the appearance that earns total eclipses the " blood moon " nickname . watchers in the eastern half of north america caught only a Sectionial eclipse – and in some places , an orange one – before the moon set below the horizon . the event started at 3:16 a.m. pt (6:16 a.m . et) , when the moon began moving into earth 's shadow . " the lunar eclipse is looking good ! " tweeted ryan hoke , a meteorologist for cnn affiliate wave in louisville , kentucky , showing a picture of a reddish Sectionial moon in a blue dawn sky . people from the u.s. west coast to australia were able to catch the total eclipse . Sections of south america , india , china and russia were able to see at least Sections of the event , but it was n't visible in greenland , iceland , europe , africa or the middle east . a lunar eclipse happens when the sun , earth and moon form a straight line in space , with the earth smack in the middle .
Reference summary: the total eclipse lasted 4 minutes and 43 seconds . people west of the mississippi river had the best view in the u.s . parts of south america , india , china and russia were able to see the eclipse
Baseline [13] the moon suddenly slipped fully into earth 's shadow at UNK a.m .et saturday , starting a total lunar eclipse for nearly five minutes . the moon entered a total lunar eclipse for nearly five minutes .
SA + syntactic unit attention mechanism: the moon slipped fully into earth 's shadow at 4:58 a.m. pacific time (7:58 a.m .et) , when the moon began moving into earth 's shadow . watchers in the eastern half of north america caught only a Sectionial eclipse .
HA-DLSTM: the moon slipped fully into earth 's shadow at 4:58 a.m. pacific time (7:58 a.m .et) saturday , what nasa says will be the shortest such eclipse of the century . watchers in the eastern half of north america caught only a Sectionial eclipse .

TABLE 7. Comparison of the human readability, relevance, and grammatical correctness on a random subset of the CNN/DM dataset.

Model Name	Readability	Relevance	Grammatical correctness
Reference	8.49	**8.91**	**10**
SA	7.53	8.12	8.1
SA-DLSTM	8.14	8.35	8.6
SA-HA-DLSTM	**8.78**	8.89	8.3

our proposed models. Human evaluators are not informed which summaries come from which model or which one is the reference summary. Each human evaluator assigns for each summary three scores between 1 to 10 to indicate its readability (how well-written the summary is), relevance (how well does the summary capture the important parts of the article), and grammatical correctness, respectively. For the grammatical correctness metrics, a summary with no grammatical error is scored 10. Each time an error is found, the score is reduced by 1, and the minimum score is 0. Our human evaluation results are shown in Table 7. The results are averaged across all examples and human evaluators.

The summary generated by our best model (SA-HA-DLSTM) achieves the best score and outperforms the reference summary on readability. The main reason maybe the length of the summary. As is shown in the example of Table 5, the reference summary consists of short and truncated sentences, whereas generated summaries by our models tend to contain more long sentences. On the other hand, by introducing Headline-Aware attention (HA), the relevance score improves 0.54 point and is comparable to that of the reference summary. The most common grammatical error is tense error. There is no significant difference in grammatical correctness for generated summaries.

V. CONCLUSION AND FUTURE WORK

We propose several improvements that address critical problems in summarization that are not adequately modeled by the basic Seq2Seq framework. We propose a syntax-augmented encoder and a headline-aware decoder. In the encoding stage, the syntactic structure information is incorporated in sentence embeddings, and the attention on syntactic units are combined with the attention on sentences to guide the summary generation. In the decoding stage, we propose a headline attention mechanism to focus on salient information in the headline. In addition, by employing Dual-memory-cell LSTM layer in the decoder, both the already generated part of summary and the content to be generated are memorized to avoid redundant content generation. The experiment results agree well with our design intention of the framework.

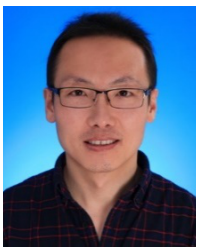
Future directions include designing a flexible data structure to accommodate syntactic parsing trees with different structures to enable batch training, and integrating the syntactic parsing model and text summarization model to achieve joint training.

REFERENCES

[1] D. R. Radev, E. Hovy, and K. McKeown, "Introduction to the special issue on summarization," *Comput. Linguistics*, vol. 28, no. 4, pp. 399–408, Dec. 2002.
[2] H. P. Luhn, "The automatic creation of literature abstracts," *IBM J. Res. Develop.*, vol. 2, no. 2, pp. 159–165, Apr. 1958.
[3] O. Tas and F. Kiyani, "A survey automatic text summarization," *Pressacademia*, vol. 5, no. 1, pp. 205–213, Jun. 2017.
[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[5] N. Zerari, M. Chemachema, and N. Essounbouli, "Neural network based adaptive tracking control for a class of pure feedback nonlinear systems with input saturation," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 1, pp. 278–290, Jan. 2019.
[6] Y. Ouyang, L. Dong, L. Xue, and C. Sun, "Adaptive control based on neural networks for an uncertain 2-DOF helicopter system with input deadzone and output constraints," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 3, pp. 807–815, May 2019.
[7] Y. Luo, S. Zhao, D. Yang, and H. Zhang, "A new robust adaptive neural network backstepping control for single machine infinite power system with TCSC," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 1, pp. 48–56, Jan. 2020.
[8] H. Lin and V. Ng, "Abstractive summarization: A survey of the state of the art," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 9815–9822.
[9] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 21, 2020, doi: 10.1109/TNNLS.2020.2979670.

[10] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
[11] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015.
[12] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 379–389.
[13] R. Nallapati, B. Zhou, C. dos Santos, C. Gulcehre, and B. Xiang, "Abstractive text summarization using Sequence-to-sequence RNNs and beyond," in *Proc. 20th SIGNLL Conf. Comput. Natural Lang. Learn.*, 2016, pp. 280–290.
[14] S. Chopra, M. Auli, and A. M. Rush, "Abstractive sentence summarization with attentive recurrent neural networks," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2016, pp. 93–98.
[15] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, 2017, pp. 1073–1083.
[16] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," 2017, *arXiv:1705.04304*. [Online]. Available: http://arxiv.org/abs/1705.04304
[17] P. Nema, M. M. Khapra, A. Laha, and B. Ravindran, "Diversity driven attention model for query-based abstractive summarization," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, 2017, pp. 1063–1072.
[18] R. Pasunuru, H. Guo, and M. Bansal, "Towards improving abstractive summarization via entailment generation," in *Proc. Workshop New Frontiers Summarization*, 2017, pp. 27–32.
[19] J. Tan, X. Wan, and J. Xiao, "Abstractive document summarization with a graph-based attentional neural model," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, 2017, pp. 1171–1181.
[20] Z. Cao, F. Wei, L. Dong, S. Li, and M. Zhou, "Ranking with recursive neural networks and its application to multi-document summarization," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 2153–2159.
[21] L. Song, Y. Zhang, Z. Wang, and D. Gildea, "A graph-to-sequence model for AMR-to-text generation," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018, pp. 1616–1626.
[22] H. Zhao, J. Cao, M. Xu, and J. Lu, "Variational neural decoder for abstractive text summarization," *Comput. Sci. Inf. Syst.*, vol. 17, no. 2, pp. 537–552, 2020.
[23] A. Celikyilmaz, A. Bosselut, X. He, and Y. Choi, "Deep communicating agents for abstractive summarization," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2018, pp. 1662–1675.
[24] S. Ma, X. Sun, W. Li, S. Li, W. Li, and X. Ren, "Query and output: Generating words by querying distributed word representations for paraphrase generation," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2018, pp. 196–206.
[25] J. Li, C. Zhang, X. Chen, Y. Cao, P. Liao, and P. Zhang, "Abstractive text summarization with multi-head attention," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–8.
[26] S. Song, H. Huang, and T. Ruan, "Abstractive text summarization using LSTM-CNN based deep learning," *Multimedia Tools Appl.*, vol. 78, no. 1, pp. 857–875, Jan. 2019.
[27] Q. Guo, J. Huang, N. Xiong, and P. Wang, "MS-pointer network: Abstractive text summary based on multi-head self-attention," *IEEE Access*, vol. 7, pp. 138603–138613, 2019.
[28] P. Kouris, G. Alexandridis, and A. Stafylopatis, "Abstractive text summarization based on deep learning and semantic content generalization," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 5082–5092.
[29] S. Gupta and S. K. Gupta, "Abstractive summarization: An overview of the state of the art," *Expert Syst. Appl.*, vol. 121, pp. 49–65, May 2019.
[30] W. S. El-Kassas, C. R. Salama, A. A. Rafea, and H. K. Mohamed, "Automatic text summarization: A comprehensive survey," *Expert Syst. Appl.*, vol. 165, Apr. 2020, Art. no. 113679.
[31] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The stanford CoreNLP natural language processing toolkit," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics, Syst. Demonstrations*, 2014, pp. 55–60.
[32] J. Cheng and M. Lapata, "Neural summarization by extracting sentences and words," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics)*, 2016, pp. 484–494.
[33] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.
[34] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems," 2015, *arXiv:1512.01274*. [Online]. Available: http://arxiv.org/abs/1512.01274
[35] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1693–1701.
[36] R. Nallapati, F. Zhai, and B. Zhou, "Summarunner: A recurrent neural network based sequence model for extractive summarization of documents," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 3075–3081.
[37] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: https://www.aclweb.org/anthology/W04-1013

JINGWEI CHENG received the Ph.D. degree from Northeastern University, China, in 2011. He is currently with the School of Computer Science and Engineering, Northeastern University. He has authored more than 40 refereed international journals and conference articles, including WI, DEXA, and IDEAL. He has also authored one monograph published by Springer. His current research interests include knowledge graphs, natural language processing, description logics, RDF data management, and ontologies.



FU ZHANG received the Ph.D. degree from Northeastern University, China, in 2011. He is currently an Associate Professor and a Ph.D. Supervisor with the School of Computer Science and Engineering, Northeastern University. He has authored more than 40 refereed international journals and conference articles. His research work was published in high-quality international conferences, such as CIKM and DEXA, and in highly cited international journals, such as fuzzy sets and systems, knowledge-based systems, and integrated computer-aided engineering. He has also authored two monographs published by Springer. His current research interests include RDF data management, ontology, knowledge graph, and knowledge representation and reasoning.

XUYANG GUO is currently pursuing the master's degree with the School of Computer Science and Engineering, Northeastern University, China. His current research interests include text summarization and knowledge graphs.