EURASIP Journal on Advances in Signal Processing

RESEARCH Open Access

Multi-task learning for abstractive text summarization with key information guide network

Weiran Xu[1*], Chenliang Li[1], Minghao Lee[2] and Chi Zhang[3]

Abstract

Neural networks based on the attentional encoder-decoder model have good capability in abstractive text summarization. However, these models are hard to be controlled in the process of generation, which leads to a lack of key information. And some key information, such as time, place, and people, is indispensable for humans to understand the main content. In this paper, we propose a key information guide network for abstractive text summarization based on a multi-task learning framework. The core idea is to automatically extract the key information that people need most in an end-to-end way and use it to guide the generation process, so as to get a more human-compliant summary. In our model, the document is encoded into two parts: results of the normal document encoder and the key information encoding, and the key information includes the key sentences and the keywords. A multi-task learning framework is introduced to get a more sophisticated end-to-end model. To fuse the key information, we propose a novel multi-view attention guide network to obtain the dynamic representations of the source text and the key information. In addition, the dynamic representations are incorporated into the abstractive module to guide the process of summary generation. We evaluate our model on the *CNN/Daily Mail* dataset and experimental results show that our model leads to significant improvements.

Keywords: Deep learning, Reinforcement learning, Text summarization, Multi-task learning, Attention mechanism

1 Introduction

Text summarization is the task of automatically generating a brief summary from a given text while maintaining the key information. There are two main approaches to text summarization: *extractive* and *abstractive*. Extractive models [1, 2] generally obtain a summary by extracting a few sentences from the original text, while abstractive models [3, 4] produce a summary by generating new sentences. Recently, the neural encoder-decoder framework [5] inspires the research on abstractive text summarization. It is generally believed that the language of this model is more fluent. Moreover, the encoder-decoder framework is also convenient for automatically adjusting the parameters.

Both the original text and the summarization are human languages. In order to generate a higher quality result, the model must be able to "understand" and represent the original text in a human-like manner. Entities such as time, place, and person are the keys for human to understand the main content. Therefore, it is essential to generate these key information into the summary. Although current abstractive models proved to be capable of capturing the regularities of the text summarization, they are hard to be controlled in the process of generation. In other words, without external guidance, it is difficult to ensure that those abstractive models could identify key information and generate them into the output [6].

Some studies have tried to solve these problems. Zhou et al. [7] proposed a selective gate network to retain

*Correspondence: xuweiran@bupt.edu.cn
[1]Beijing University of Posts and Telecommunications, Xitucheng Road, Beijing 100876 , China
Full list of author information is available at the end of the article

more key information in the summary. However, the selective gate network, which is controlled by the representation of the input text, controls the information flow from encoder to decoder for just once. If some key information does not pass the network, it is hard for them to appear in the summary. See et al. [8] proposed a pointer-generator model, which uses the pointer mechanism [9] to copy words from the input text, to deal with the out-of-vocabulary (OOV) words. Without external guidance, it is hard for the pointer to identify keywords. In previous work, we combine the extractive model and the abstractive model and use the former one to obtain keywords as guidance for the latter one [10]. However, this model is not sophisticated enough. And it is a pipelined system that extracts keywords by the TextRank algorithm.

In addition, the decoder model, especially the attention mechanism, is also crucial for generating a summary containing all the key information. Hsu et al. [11] proposed a unified model which combines sentence-level and word-level attentions by simple scalar multiplication and renormalization. Their sentence-level attention is fixed, which means it will be the same for each generated word. However, the focus of the abstractive model on sentence should be constantly changing during the summary generation. Tan et al. [6] proposed a graph-based attention method that can discover the salient information of a document in the encoder-decoder framework. They also use the TextRank algorithm to identify key sentences in the input text. We argue that the abstractive summarization model should contain key information extraction and then use the key information to dynamically guide the process of summary generation.

In this paper, we propose a key information guide network for abstractive text summarization based on a multi-task learning framework. Our core idea is to automatically extract the key information that people need most in an end-to-end way and use it to guide the generation process, so as to get a more human-compliant summary. In our model, the document is coded into two parts: the result of the document encoder and the key information encoding, and the key information includes the key sentences and the keywords. Multi-task learning framework is introduced to get a more sophisticated end-to-end model. The main part is an abstractive model based on encoder-decoder structure, in which a normal document encoder is employed. The second task is the key sentence extraction, and an extractive method is included here. Another extractive model is used to extract keywords. The extractive models and the abstractive model will be trained jointly in this multi-task framework, so they can benefit each other. In this encoder model, several semantic coding layers are naturally formed, i.e., word layer, sentence layer, and document layer. In order to simplify the model, our key information, i.e., key sentences and keywords, is extracted from the outputs of sentence layer encoder. In the decoder, the key information will guide the generation process by two ways: the attention mechanism and the pointer mechanism. To fuse the key information, we propose a novel multi-view attention guide network to obtain the dynamic representations of the source text and the key information. In addition, the dynamic representations are incorporated into the abstractive module to guide the process of summary generation. Experiments show that our model achieves significant improvements.

Our contributions are as follows:

- We propose a key information guidance network model to obtain more human-compliant summaries. In this model, a document is represented as keyword encoding, key sentence encoding, and document encoding, and then, they work together to guide the generation of the summary.
- Multi-task learning framework is introduced to get a more sophisticated end-to-end model. Extractive models and the abstractive model are fused in an end-to-end model, which can make full use of various training data to adjust model parameters.
- In the decoder, we propose a novel multi-view attention guide network to obtain the dynamic representations of the source text and the key information. In addition, the dynamic representations are incorporated into the abstractive module to guide the process of summary generation.

The rest of the paper is structured as below: Section 2 will review the related work. Section 3 introduces the key information guide network, which is our baseline. Section 4 presents our detailed model description, followed by experiments and analysis in Section 5. At last, we conclude our work in Section 6.

2 Related work

2.1 Abstractive summarization

Since Rush et al. [3] first bring up the encoder-decode framework in the task of text summarization, abstractive models [6, 10, 12] have been widely used to generate the summary like human being. Hsu et al. [11] propose a unified model combining sentence-level and word-level attentions to take advantage of both extractive and abstractive summarization approaches. Most methods select content at the sentence level [11] and the word level [10]. Current state-of-the-art models use the pointer-generator mechanisms. Our model selects the key information including keywords and key sentences and then incorporates the key information in the abstractive module to guide the process of summary generation.

2.2 Extractive summarization

Yasunaga et al. [2] use a graph-based neural model to extract salient sentences. Nallapati et al. [4] use recurrent neural networks to read the article and get the representations of the sentences and article to select sentences. Most models utilize side information (i.e., image captions and titles) to help the sentence classifier choose sentences. In addition, Tan et al. [6] combined recurrent neural networks with graph convolutional networks to compute the salience (or importance) of each sentence. Neural models are able to leverage large-scale corpora and achieve better performance than traditional methods. While some extractive summarization models are able to achieve better performance, they all suffer from low relevance between sentences.

2.3 Content selection

Since text summarization requires key information selection, the abstractive models first select key information and then generate the summary based on it. Some recent work solves the sentence extraction and document modeling in an end-to-end framework. Nallapati et al. [4] propose an encoder-decoder approach where the encoder hierarchically learns the representation of sentences and documents while an attention-based sentence extractor extracts salient sentences sequentially from the original document. Then, based on the above work, they propose a recurrent neural network-based sequence-to-sequence model for sequential labeling of each sentence in the document. However, some key information may be lost in the selection process, which leads to an inability of the generation.

2.4 Pointer-generator network

The pointer network [9] is a sequence-to-sequence model, which uses a soft attention distribution to produce an output sequence consisting of elements in the input sequence. Pointer networks have been used to create a hybrid approach to NMT (neural machine translation), language models, and abstractive models. This method is close to the mandatory note compression model and the CopyNet model. The pointer-generator model [8] has some small differences: they calculate the explicit switching probability to copy words from the input text. In addition, the recycling of attention distribution is distributed as a copy, which decides whether to generate a word from the vocabulary or the input text. When multiple words appear in the source text, we obtain the probability mass of all corresponding parts of the attention distribution. They believe that calculating explicit switch probability is useful for improvement. In this way, they reduce the probability of all generated words or all copied words at once, rather than reducing them separately. These two distributions are very useful, and they find that their simpler approach is sufficient for similar purposes. From the experimental results, they observe that the pointer mechanism often copies a word and multiple occurrences of the word appear in the source text.

The pointer-generator model is very different from the method of other pointer network. These jobs train their pointer components to activate only vocabulary words or named entities, and they do not mix the probability of replicating distributions and lexical distributions. The pointer-generator model proposes a hybrid approach described here and is more suitable for abstract summarization. In addition, they demonstrate that replication mechanisms are critical for accurately replicating rare but vocabulary words.

2.5 Prediction-guide mechanism

He et al. [13] propose a prediction network to predict the long-term value in the final generation summary. In addition, they propose to use a prediction network to improve beam search, which takes the source sentence, the currently available decoded output, and the candidate words at step t as inputs and predicts the long-term value of the partial target sentence (e.g., the BLEU score) if it is done by the NMT (neural machine translation) model. In accordance with the practice of reinforcement learning, they call it the network value network. Specifically, they propose a recurring structural network of values and train their parameters from bilingual data. During the test time, when selecting the word w for decoding, they consider the conditional probability given by the NMT model and the long-term value predicted by the value network. Our forecasting guidance mechanism is used to ensure more critical information covered in the final summary.

2.6 Multitask learning and joint training

Multi-task learning and joint training are also unavoidable problems in this paper. However, this paper mainly refers to [14–22] to construct our own algorithm, which cannot be regarded as a new model.

3 The key information guide network

In this section, we will introduce the key information guide network (KIGN) [10] (Fig. 1), which is served as our baseline.

3.1 Encoder-decoder model based attention

Our encoder-decoder framework is similar to that of [4]. The tokens of the input article $x = \{w_1, w_2, ..., w_n\}$ are entered into the encoder, which maps the text into the encoder hidden state sequence $\{h_1, h_2, ..., h_n\}$. At each decoding time step t, the previous word embedding w_{t-1} and the previous context vector c_{t-1} as input to obtain the decoder hidden state s_t. The context vector c_t is obtained

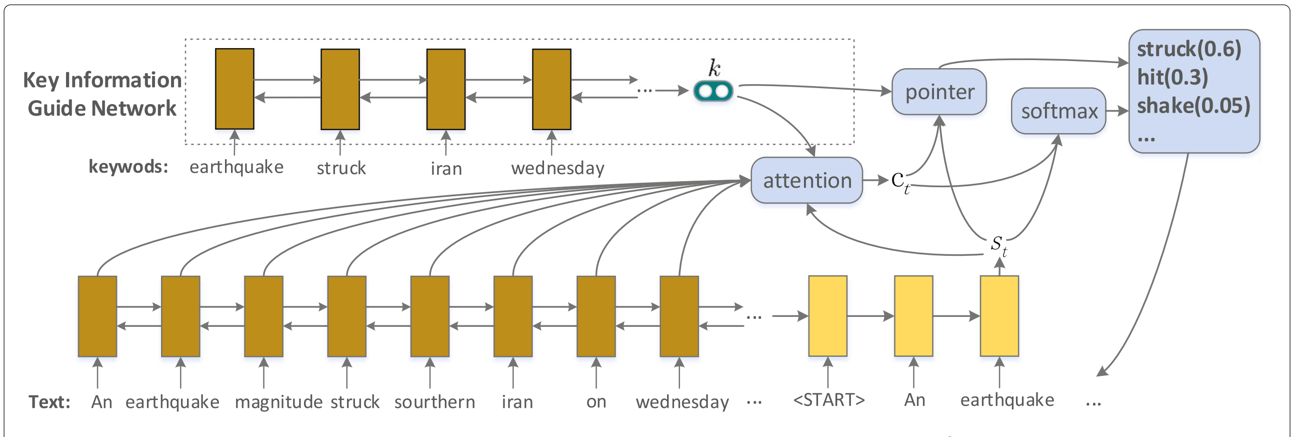


Fig. 1 The key information guide network. In the key information guide network, we encode the keywords to the key information representation [10]

by using the attention mechanism :

$$e_{ti} = v^T \tanh(W_h h_i + W_s s_t) \tag{1}$$

$$\alpha_t^e = \text{softmax}(e_t) \tag{2}$$

$$c_t = \textstyle\sum_{i=1}^{n} \alpha_{ti}^e h_i \tag{3}$$

where v, W_h, and W_s are learnable parameters and h_i is the hidden state of the input token w_i.

The context vector c_t, which represents what has been read from the source text, is concatenated with the decoder hidden state s_t to predict the next word with a softmax layer over the whole vocabulary:

$$P(y_t|y_1, ..., y_{t-1}) = \text{softmax}(f(s_t, c_t)) \tag{4}$$

where f represents a linear function or a neural network.

3.2 Key information guide network

Most encoder-decoder models [7, 8] simply take the source text as input and then output a summary, which is difficult to control during the generating process, resulting in missing key information in the summary. We propose a key information-guided network which can guide the generation process by the attention mechanism and the pointer mechanism.

Firstly, keywords are extracted by TextRank algorithm. Then, a max-pooling CNN model [23, 24] is employed to merge the key information. As shown in Fig. 1, the keywords are entered one by one into the key information guide network, and then, we connect the last forward hidden state $\overrightarrow{h_n}$ and the backward hidden state $\overleftarrow{h_1}$ as key information representation k:

$$k = \begin{bmatrix} \overleftarrow{h_1} \\ \overrightarrow{h_n} \end{bmatrix} \tag{5}$$

The traditional attention mechanism is difficult to identify keywords, which only use the decoder state as a query to obtain the attention distribution of the encoder hidden state. We use the key information to represent k as an extra input to the attention mechanism and change Eq. (1) to:

$$e_{ti} = v^T \tanh(W_h h_i + W_s s_t + W_k k) \tag{6}$$

where W_k is a learnable parameter. We use the new e_{ti} to obtain new context vector c_t (Eq. 3) and attention distribution α_t^e (Eq. 2) .

Then, we apply k to represent the key information and use the new context vector c_t to calculate the probability distribution of all words in the vocabulary, changing Eq. (4) to:

$$P_v(y_t|y_1, ..., y_{t-1}) = \text{softmax}(f(s_t, c_t, k)) \tag{7}$$

where v represents that y_t is from the target vocabulary.

The KIGN makes the attention mechanism more focus on the keywords, which is similar to introduce prior knowledge to the model.

3.3 Pointer mechanism

In order to deal with the OOV (out-of-vocabulary) problem, we combine the pointer network [9] with our key information-based generating method, which enable us to copy words as well as generate text. In the pointer generator model, we need to compute a soft switch p_{sw} to select between the generated words and the duplicated words:

$$p_{sw} = \sigma(w_k^T k + w_c^T c_t + w_{s_t}^T s_t + b_{sw}) \tag{8}$$

where w_k^T, w_c^T, w_s^T, and b_{sw} are parameters and σ is the sigmoid function.

Our pointer mechanism is equipped with a key information representation that identifies the keyword. We use the new attention distribution α_{ti}^e as the probability of entering the token w_i and obtain the following probability distribution to predict the next word:

$$P(y_t = w) = p_{sw} P_v(y_t = w) + (1 - p_{sw}) \sum_{i:w_i=w} \alpha^e_{ti} \quad (9)$$

Note that if w is an OOV word, $P_v(y_t = w)$ is zero.

In the process of training, we train the model by minimizing the maximum likelihood loss in each decoding time step t, which is the most widely used in the task of generation. In addition, we define y^*_t as the target word for the decoding time step t, and the overall loss is:

$$L = -\frac{1}{T} \sum_{t=0}^{T} \log P(y^*_t | y^*_1, ..., y^*_{t-1}, x) \quad (10)$$

4 Multi-task learning for abstractive text summarization with KIGN

The KIGN introduced in Section 3 enables the text summary generator to pay more attention to the key information that humans are most concerned about. However, this model is not sophisticated enough. For example, key information is obtained through the Textrank algorithm rather than a learning-based approach. In this section, we propose a multi-task learning model (Fig. 2) for abstractive text summarization based on the KIGN framework, which includes a novel document encoder, a key information extractor, and some methods such as joint training and prediction-guide mechanism.

As shown in Fig. 2, firstly, we propose a document encoder which includes a word-level encoder and a sentence-level encoder. In this way, we can obtain global features for each word and each sentence's encoding respectively. Then, the key information extraction layer selects keywords and key sentences. Next, a multi-source attention will guide the process of generation. Finally, the abstracter and key information extractors, which includes a keyword extractor and a key sentence extractor, are trained by minimizing three loss functions in an end-to-end manner.

4.1 Document encoder

We propose a novel document encoder to encode words and sentences respectively instead of only using the hierarchical encoder to encode both [4]. This method will facilitate the subsequent keyword extraction and key sentence extraction.

4.1.1 The global word encoder

The task of keyword extraction requires the information of the whole document as well as the information of one single word. In order to encode global information into a hidden state to represent a word, the encoding of each word must be based on at least one layer of neural network. Therefore, we use a bidirectional LSTM as the global word encoder. The tokens of the input text $\{w_1, w_2, ..., w_n\}$ are forward fed into the global word encoder, which maps the text into a sequence of hidden states $\{\overrightarrow{h^w_1}, \overrightarrow{h^w_2}, ..., \overrightarrow{h^w_n}\}$:

$$\overrightarrow{h^w_i} = \text{BiLSTM}(\overrightarrow{h^w_{i-1}}, e^w_i) \quad (11)$$

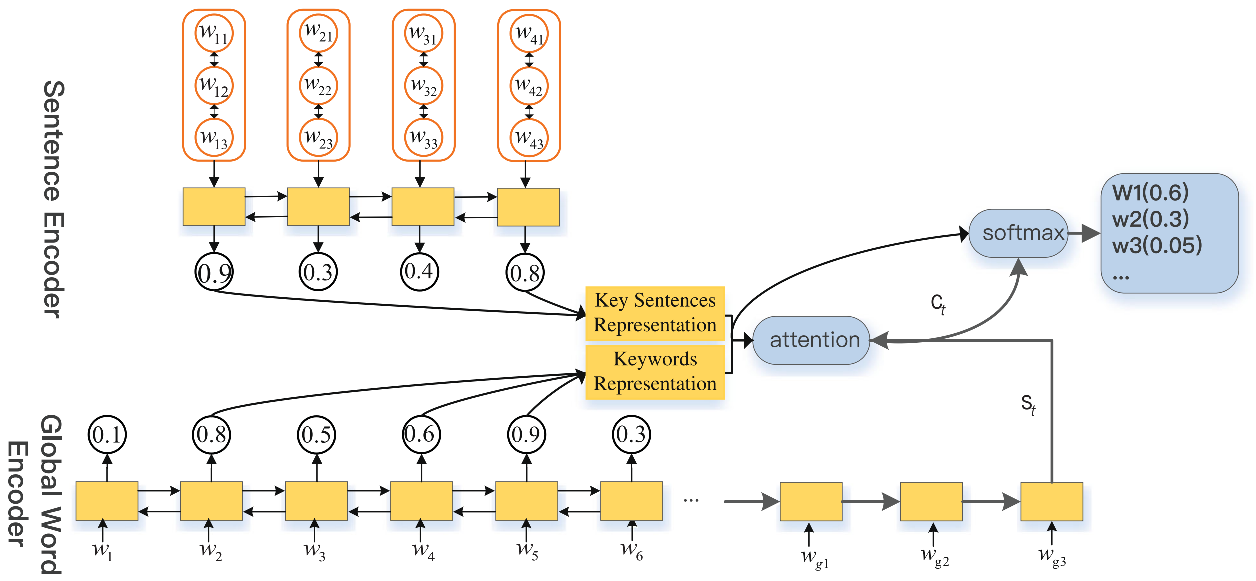


Fig. 2 Architecture of the document encoder and key information extraction. The document encoder includes a global word encoder and a sentence encoder. Key information extraction consists of keyword extractor and key sentence extractor, and then two kinds of key information representation are obtained

where $\overrightarrow{h_i^w}$ and e_i^w denote the forward hidden state and embedding of word w_i, respectively, and $i = 1, \cdots, n$. Likewise, backward hidden states can also be obtained, i.e., $\overleftarrow{h_i^w}$ $i = 1, \cdots, n$. Therefore, the encode of the ith word is:

$$h_i^w = \begin{bmatrix} \overleftarrow{h_i^w} \\ \overrightarrow{h_i^w} \end{bmatrix} \tag{12}$$

We concat the forward and backward final states in the global word encoder to obtain the representation of the document:

$$d_w = \begin{bmatrix} \overleftarrow{h_1^w} \\ \overrightarrow{h_n^w} \end{bmatrix} \tag{13}$$

4.1.2 *The sentence encoder*

The sentence encoder is a hierarchical encoder, which consists of the word-level encoder and the sentence-level encoder. A document can be seen as a sequence of words $\{w_1, w_2, ..., w_n\}$. And this document also can be seen as a sequence of sentences $\{s_1, s_2, ..., s_m\}$ and $s_i = \{w_{i,1}, w_{i,2}, \cdots\}$. As shown in Fig. 2, the word-level encoder is a bidirectional LSTM that encodes the words of each sentence into the sentence representation:

$$h_{i,j}^w = \text{BiLSTM}(\overrightarrow{h_{i,j-1}^w}, \overleftarrow{h_{i,j+1}^w}, e_{i,j}^w) \tag{14}$$

where $h_{i,j}^w$ and $e_{i,j}^w$ denotes the hidden state and embedding of word $w_{i,j}$, respectively. We concat the forward and backward final hidden states in the word-level encoder as the sentence representation:

$$x_i^s = \begin{bmatrix} \overleftarrow{h_{i,1}^w} \\ \overrightarrow{h_{i,n_i}^w} \end{bmatrix} \tag{15}$$

Then, we use another bidirectional LSTM as the sentence-level encoder, which updates each sentence representation:

$$h_i^s = \text{BiLSTM}(\overrightarrow{h_{i-1}^s}, \overleftarrow{h_{i+1}^s}, x_i^s) \tag{16}$$

where h_i^s denotes the hidden state of sentence s_i. The representation of the document based on the sentence encoder is:

$$d_s = \begin{bmatrix} \overleftarrow{h_1^s} \\ \overrightarrow{h_m^s} \end{bmatrix} \tag{17}$$

4.2 Key information extraction

The task of text summarization needs to remove the unnecessary information and retain key information from the input document. Since it is difficult for the encode-decoder framework to find keywords and key sentences, we employ a key information extraction model to extract keywords and key sentences. Key information extractors can be trained by supervised learning in a multi-task framework.

4.2.1 *Keyword extraction*

First, we use the results of the global word encoder, i.e., h_i^w $i = 1, \cdots, n$, and d_w, to extract keywords in the sequence to sequence model:

$$g_i^w = \sigma(W_g h_i^w + U_g d_w + b_g) \tag{18}$$

where W_g and U_g denote weight parameters, b_g the bias vector, and σ the sigmoid activation function. Then, the top N words $\{w_1^k, ..., w_N^k\}$ are obtained as the keywords.

4.2.2 *Key sentence extraction*

Similarly, the codes based on the sentence encoder, i.e., h_i^s $i = 1, \cdots, m$, and d_s, are used to extract key sentences. For each sentence s_i, the gate network takes the d_s and h_i^s as inputs to calculate the probability value:

$$g_i^s = \sigma(W_s h_i^s + U_s d_s + b_s) \tag{19}$$

where W_s and U_s denote weight matrices, b_s the bias vector, and σ the sigmoid activation function. We also regard the top M sentences $\{s_1^k, ..., s_M^k\}$ as the key sentences based on the probability value of each sentence.

4.3 Improved key information guide network

Based on the abovementioned methods, the KIGN introduced in Section 3 can be improved. The first is to update the presentation of key information. Now, two kinds of key information are obtained, i.e., keywords and key sentences. According to Section 3, we can get keyword representation k_w and key sentence representation k_s, and k_w and k_s are substitutes for k.

The second is to update the document representation. In Section 4.1, the document is encoded into the hidden states of words, i.e., $\{h_i^w | i = 1, \cdots, n\}$, and the hidden states of sentences, i.e. ,$\{h_i^s | i = 1, \cdots, m\}$. By replacing h_i in (1)(3) with h_i^w, the word encoder-based context vector c_t^w is obtained:

$$e_{ti}^w = v^T \tanh(W_h^w h_i^w + W_s s_t + W_{k_w} k_w + W_{k_s} k_s) \tag{20}$$

$$c_t^w = \sum_{i=1}^{n} \alpha_{ti}^{e,w} h_i^w \tag{21}$$

In the same way, the sentence encoder based context vector c_t^s can also be obtained. As a result, (7) is updated by:

$$P_v(y_t | y_1, ..., y_{t-1}) = \text{softmax}(f(s_t, c_t^w, c_t^s, k_w, k_s)) \tag{22}$$

In addition, there are still some trivial parts that need to be updated, such as (8), which will not be repeated here.

4.4 Joint training the model

We jointly train the abstract generator, keyword extractor, and key sentence extractor by minimizing three loss functions: L_{abs}, L_{kw}, and L_{ks}. The final loss is as below:

$$L_{s2s} = \lambda_1 L_{abs} + \lambda_2 L_{kw} + \lambda_3 L_{ks} \tag{23}$$

where λ_1, λ_2, and λ_3 are hyper-parameters.

Similar to [8], we use the pointer mechanism to calculate the final word distribution $P^{\text{final}}(\hat{y}_t)$. Then, we train the abstracter by minimizing the negative log-likelihood:

$$L_{abs} = -\frac{1}{T}\sum_{t=1}^{T} \log P^{\text{final}}(\hat{y}_t) \tag{24}$$

where $\hat{y}_t$ is the tth token in the reference summary.

In our model, the losses of the keyword extraction L_{kw} and the key sentence extraction L_{ks} are the binary cross entropy:

$$L_{kw} = -\frac{1}{N}\sum_{i=1}^{N}(r_i^w \log g_i^w + (1-r_i^w)\log(1-g_i^w)) \tag{25}$$

$$L_{ks} = -\frac{1}{M}\sum_{i=1}^{M}(r_i^s \log g_i^s + (1-r_i^s)\log(1-g_i^s)) \tag{26}$$

where M, N are hyper-parameters.

Ground truth label. To obtain the keywords, we filter the stop words in the reference summary and use the remaining as keyword ground truth labels $r^w = \{r_i^w\}_i$. For the key sentence ground truth label $r^s = \{r_i^s\}_i$, we measure the informativity of each sentence in the text and select key sentences similar to [11]. In order to obtain the ground truth label, we first measure the informativeness of each sentence in the article by calculating the ROUGE-L recall score between the sentence and the reference abstract summary. Second, we sort the sentences according to the amount of information and select sentences in order of information from high to low. If the new sentence can increase the amount of information for all selected sentences, we add a sentence at a time. Finally, we obtained the ground truth label and trained our extractor by minimizing the loss function. Our method is similar to [11] that aims to extract the final summary of the article so that they use the ROUGE F-1 score to select the real sentence. In our model, we use the ROUGE recall score to get as much reference summary information as possible.

4.5 Prediction-guide mechanism at test time

In the process of test time, when the model predict the next word, we not only consider the above probability (Eq. 9), but also consider the long-term value predicted by the prediction guidance mechanism. The predictive guidance mechanism is based on [13].

Our predictive guidance mechanism is a single layer feedforward network with the sigmoid activation function that predicts the range of critical information covered in the final summary. At each decoding time step t, we perform a mean summation of the decoder hidden state. $\bar{s}_t = \frac{1}{t}\sum_{l=1}^{t} s_l$, the encode states $\bar{h}_n = \frac{1}{n}\sum_{i=1}^{n} h_i$, and the key information k as inputs to obtain the long-term value.

We sample the two parts for each x summary y_{p1} and y_{p2} and randomly stop to get $\bar{s}_t$. Then, we complete the build from y_p to get the M average decoder hidden state $\bar{s}$ completed summary $S(y_p)$(using beam search) and calculate the average score:

$$\text{AvgCos}(x, y_p) = \frac{1}{M}\sum_{\bar{s}\in S(y_p)} \cos(\bar{s}, k) \tag{27}$$

where cos is the function of cosine similarity.

We hope the predicted value of $v(x, y_{p1})$ can be larger than $v(x, y_{p2})$ if $\text{AvgCos}(x, y_{p1}) > \text{AvgCos}(x, y_{p2})$. Therefore, the loss function of the predicted boot network is as follows:

$$L_{pg} = \sum_{(x, y_{p1}, y_{p2})} e^{v(x,y_{p2}) - v(x,y_{p1})} \tag{28}$$

where $\text{AvgCos}(x, y_{p1}) > \text{AvgCos}(x, y_{p2})$.

In testing, we first calculate the normalized logarithm probability for each candidate and then linearly combine it with the predicted guidance network predictive values. In addition, given the abstract model $P(y|x)$ (Eq. 9), predict the boot network $v(x, y)$ and the hyperparameter $\alpha \in (0, 1)$. The score of the sequence y of x is calculated by:

$$\alpha \times \log P(y|x) + (1-\alpha) \times \log v(x, y) \tag{29}$$

where $\alpha \in (0, 1)$ is a hyperparameter.

5 Experiments

5.1 Experiment setup

The *CNN/Daily Mail* dataset [4, 25] is employed here, and the data is processed in the same way as [8]. We use three 300-dimensional LSTMs for the global word encoder and sentence encoder. And a 50k-word vocabulary is used. During the training and testing, we truncate the text to 400 tokens for word encoder and limit the length of the summary to 100 tokens. We train the model using Adagrad [15] with learning rate 0.15 and an initial accumulator value of 0.1. The batch size is set to 16, and the number of keywords and key sentences are 40 and 10. We jointly train the three tasks and set $\lambda_1 = 1$ and $\lambda_2 = \lambda_3 = 0.5$. Following the previous work, our major evaluation metric is F-score of ROUGE.

In addition, for the prediction mechanism, we use the single-layer feed forward network and set the number of nodes with 800. For the hyperparameter α, we use a different α to test the performances of KIGN+prediction-guide model during the decoding time. As can be seen from Fig. 3, the performance of our model is stable for the α ranging from 0.8 to 0.95. In addition, when we set the α as 0.9, we can get the highest F-score. We can see that when we set M as 8 and adapt mini-batch training with the batch size of 16. The network is trained with AdaDelta.

During the process of the training and testing, we truncated the input token to 400 and limited the length of the

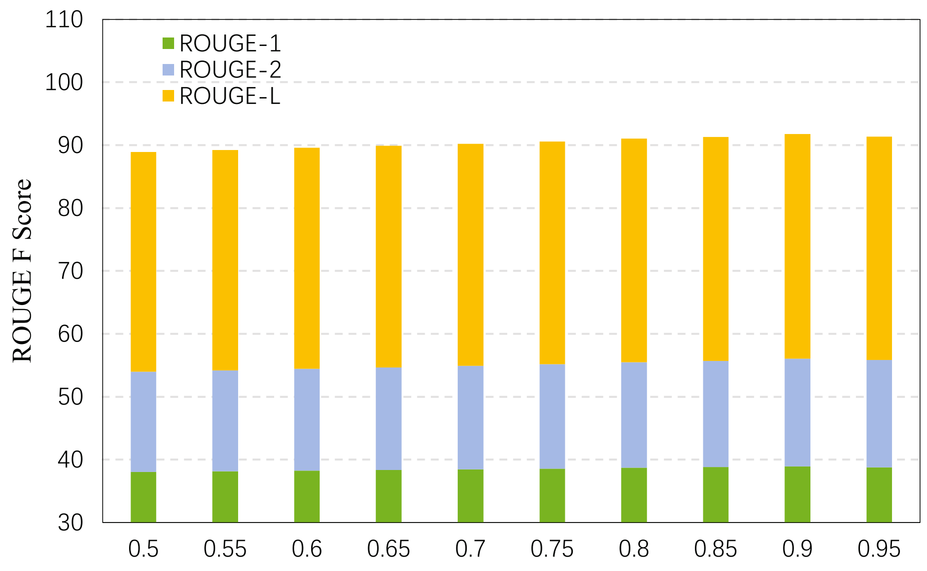


Fig. 3 The performance of our model. ROUGE-1, ROUGE-2, and ROUGE-L F1 scores of KIGN+prediction-guide model w.r.t different hyperparameter α

output summary to 100 tokens. Similar to [8] at the time of the test, we truncated the input token to 400 and limited the length of the output summary to 120 tokens. We trained the keyword network model with training iterations of less than 200,000. Then, we train a single-layer feed forward network based on the KIGN model. Finally, during the test, we combine the KIGN model with the predictive guidance mechanism to generate a summary.

5.2 Results and discussion

The experimental results are shown in Table 1. The first five are commonly used sequence to sequence methods: Seq2Seq model enhanced by attention mechanism with a 150k vocabulary, Seq2Seq model enhanced by attention mechanism with a 50k vocabulary, Seq2Seq model with a graph-attention, hierarchical attention networks method [4], and Seq2Seq model equipped with pointer-mechanism.

Table 1 shows that our baseline model, named key information guide network (shown in Fig. 1), obtain better scores than Seq2Seq model equipped with pointer-mechanism by + 1.3 ROUGE-1, + 0.9 ROUGE-2, and + 1.0 ROUGE-L. With the help of the pointer-mechanism (shown in Fig. 1), key information guide network (KIGN+Prediction-guide) has achieved much better results by + 2.5 ROUGE-1, + 1.5 ROUGE-2, and + 2.2 ROUGE-L, while Key Information Guide Network with a multi-task learning frame (shown in Fig. 2) obtains the best scores, and the additional improvements are + 0.2 ROUGE-1, + 0.2 ROUGE-2, + and 0.2 ROUGE-L. We can also see in Table 1 that if the keywords and sentences are given, the result can be better (40.34 ROUGE-1,

Table 1 ROUGE F1 scores for models on the CNN/Daily Mail test set

Model	ROUGE-1	ROUGE-2	ROUGE-L
Seq2Seq + attention (150k vocab)	30.49	11.17	28.08
Seq2Seq + attention (50k vocab)	31.33	11.81	28.83
Graph-attention	38.1	13.9	34.0
Hierarchical attention networks	35.46	13.30	32.65
Seq2Seq with pointer mechanism	36.44	15.66	33.42
Key information guide network	37.76	16.56	34.49
KIGN+prediction-guide	38.95	17.12	35.68
Our model (joint training)	**39.15**	**17.34**	**35.92**
Our model (given keywords and key sentences)	40.34	17.70	36.57

All our ROUGE scores have a 95% confidence interval of at most ± 0.25 as reported by the official ROUGE script

17.70 ROUGE-2, 36.57 ROUGE-L), which also proves that it is reasonable to use the key information to guide the generation of text summarization.

5.3 Case study

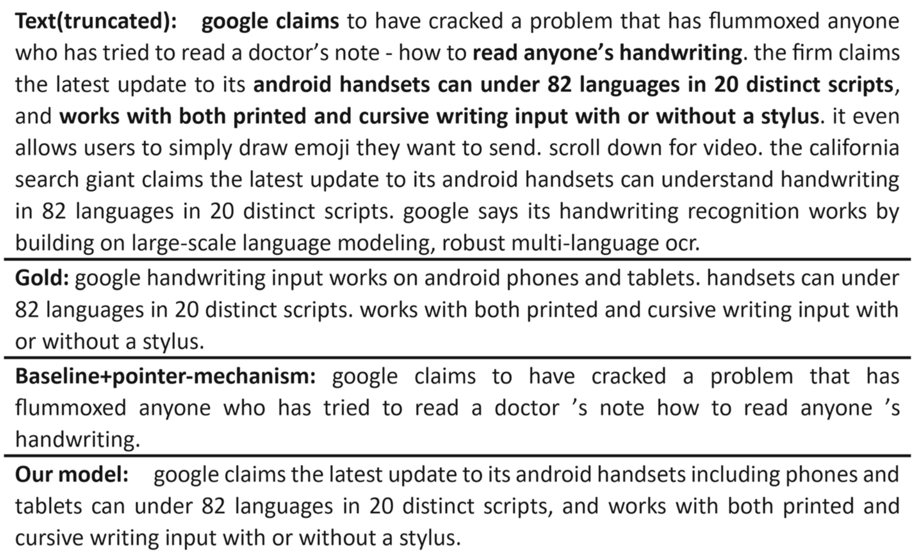
In order to demonstrate the ability of our method to obtain key information, the processing results of a specific piece of text are shown in Fig. 4. The original is listed in the top half of Fig. 4, and the key information is identified in bold. Next are the given gold summary and the outputs of the two models. The original text is about Google handwriting input working on android handsets and some function introduction, and the key information is "google claims," "read anyone's handwriting," "android handsets can under 82 languages in 20 distinct scripts," and "works with both printed and cursive writing input with or without a stylus." We can see that the summary of the baseline model equipped with pointer-mechanism covers only "google have cracked the problem of reading handwriting." While our model summarizes almost all the key information.

5.4 Ablation studies

We conduct ablation experiments to show the effect of our multi-view attention guide network. As show in Table 2, we can observe that all kinds of view contribute more or less performance boost to the model. Apparently, we can observe that the keyword view attention and guide generation makes much contribution to the improvement of our model. First, we can observe that the key information extraction can improve the scores of ROUGE (+ 1.1 ROUGE-1, + 0.6 ROUGE-2, + 0.9 ROUGE-L). Second, the prediction guide can improve the scores of ROUGE (+ 0.8 ROUGE-1, + 0.4 ROUGE-2, + 0.5 ROUGE-L). Then, the guide generation can improve the scores of ROUGE (+ 1.4 ROUGE-1, + 0.9 ROUGE-2, + 1.3 ROUGE-L). In addition, the guide pointer can improve the scores of ROUGE (+ 0.4 ROUGE-1, + 0.2 ROUGE-2, + 0.3 ROUGE-L). Finally, the joint training can improve the scores of ROUGE (+ 1.9 ROUGE-1, + 1.1 ROUGE-2, + 1.5 ROUGE-L). From the results, we can observe that our joint training can significantly improve the score of ROUGE.

6 Conclusions

In this paper, we propose a multi-task learning model with key information guide network that combines the extractive method and the abstractive method in a novel way. Our model is based on the key information guide network. The key information guide network uses an extractive method to obtain the keywords from the text. Then, it encodes the keywords to the key information representation and integrates it into the abstractive model to guide the process of generation. The guidance is mainly in two aspects: the attention mechanism and the pointer mechanism. Based on the key information guide network, we propose a multi-task learning model to jointly train the extractive model and the abstractive model. Specifically, we use a document encoder to encode words and sentences of the input text respectively. Then, we extract key information including keywords and key sentences based on the encoder. In this way, our key information extraction is not from the TextRank method. We obtain the key information from our sequence to


Text(truncated): **google claims** to have cracked a problem that has flummoxed anyone who has tried to read a doctor's note - how to **read anyone's handwriting**. the firm claims the latest update to its **android handsets can under 82 languages in 20 distinct scripts**, and **works with both printed and cursive writing input with or without a stylus**. it even allows users to simply draw emoji they want to send. scroll down for video. the california search giant claims the latest update to its android handsets can understand handwriting in 82 languages in 20 distinct scripts. google says its handwriting recognition works by building on large-scale language modeling, robust multi-language ocr.

Gold: google handwriting input works on android phones and tablets. handsets can under 82 languages in 20 distinct scripts. works with both printed and cursive writing input with or without a stylus.

Baseline+pointer-mechanism: google claims to have cracked a problem that has flummoxed anyone who has tried to read a doctor 's note how to read anyone 's handwriting.

Our model: google claims the latest update to its android handsets including phones and tablets can under 82 languages in 20 distinct scripts, and works with both printed and cursive writing input with or without a stylus.

Fig. 4 Comparison of the output of two models on a news article. Bold words in text are the key information (baseline: enc-dec+attn; our model: KIGN+prediction-guide)

Table 2 Ablation studies of our model on the test set

Model	ROUGE-1	ROUGE-2	ROUGE-L
Our model	**39.15**	**17.34**	**35.92**
W/o prediction-guide	38.16	17.17	34.95
W/o key information extraction	37.25	16.15	34.19
W/o pointer network	37.83	16.52	34.68
W/o joint training	36.35	15.62	33.38

sequence model. Finally, we jointly train the three tasks of keyword extraction, key sentence extraction, and summary generation. At test time, we use a prediction-guide mechanism, which can obtain the long-term value for future decoding, to further guide the summary generation. Experiments show that our model leads to significant improvements.

Abbreviations
OOV: Out-of-vocabulary; NMT: Neural machine translation; KIGN: Key information guide network

Acknowledgements
We thank the reviewers for useful comments. This work was supported by Beijing Natural Science Foundation (4174098), National Natural Science Foundation of China (61702047), National Natural Science Foundation of China (61703234), and the Fundamental Research Funds for the Central Universities (2017RC02).

Authors' contributions
LCL put forward the original idea, and then XWR and LCL improved the method together. LCL performed the algorithms and experiments, and XWR wrote the final paper. LMH and ZC provided many valuable advices. The authors read and approved the final manuscript.

Funding
This work was supported by National Key R&D Program of China No. 2019YFF0303300 and Subject II No. 2019YFF0303302, DOCOMO Beijing Communications Laboratories Co., Ltd, and MoE-CMCC "Artifical Intelligence" Project No. MCM20190701.

Availability of data and materials
Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

Consent for publication
Not applicable. No personal privacy data has been published.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Beijing University of Posts and Telecommunications, Xitucheng Road, Beijing 100876 , China. [2]China General Technology Research Institute, Beijing, China. [3]Energy Processes Division Royal Institute of Technology, Teknikringen 42, SE-100 44 Stockholm, Sweden.

Received: 28 October 2019 Accepted: 10 March 2020
Published online: 19 April 2020

References
1. R. Mihalcea, P. Tarau, in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Text rank: bringing order into text (Association for Computational Linguistics, Barcelona, Spain, 2004), pp. 404–411. https://www.aclweb.org/anthology/W04-3252
2. M. Yasunaga, Z. Rui, K. Meelu, A. Pareek, D. Radev, Graph-based neural multi-document summarization. CoRR. **abs/1706.06681** (2017). 1706.06681
3. A. M. Rush, S. Chopra, J. Weston, in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. A neural attention model for abstractive sentence summarization (Association for Computational Linguistics, Lisbon, Portugal, 2015), pp. 379–389. https://www.aclweb.org/anthology/D15-1044. https://doi.org/10.18653/v1/D15-1044
4. R. Nallapati, B. Xiang, B. Zhou, Sequence-to-sequence rnns for text summarization. CoRR. **abs/1602.06023** (2016). 1602.06023
5. I. Sutskever, O. Vinyals, Q. V. Le, in *Advances in Neural Information Processing Systems 27*. Sequence to sequence learning with neural networks (Curran Associates, Inc., 2014), pp. 3104–3112. http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf
6. J. Tan, X. Wan, J. Xiao, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Abstractive document summarization with a graph-based attentional neural model (Association for Computational Linguistics, Vancouver, Canada, 2017), pp. 1171–1181. https://www.aclweb.org/anthology/P17-1108. https://doi.org/10.18653/v1/P17-1108
7. Q. Zhou, N. Yang, F. Wei, M. Zhou, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Selective encoding for abstractive sentence summarization (Association for Computational Linguistics, Vancouver, Canada, 2017), pp. 1095–1104. https://www.aclweb.org/anthology/P17-1101. https://doi.org/10.18653/v1/P17-1101
8. A. See, P. J. Liu, CD. Manning, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Get to the point: summarization with pointer-generator networks (Association for Computational Linguistics, Vancouver, Canada, 2017), pp. 1073–1083. https://www.aclweb.org/anthology/P17-1099. https://doi.org/10.18653/v1/P17-1099
9. O. Vinyals, M. Fortunato, N. Jaitly, in *Advances in Neural Information Processing Systems 28*, ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Pointer networks (Curran Associates, Inc., 2015), pp. 2692–2700. http://papers.nips.cc/paper/5866-pointer-networks.pdf
10. C. Li, W. Xu, S. Li, S. Gao, in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Guiding generation for abstractive text summarization based on key information guide network (Association for Computational Linguistics, New Orleans, Louisiana, 2018), pp. 55–60. https://www.aclweb.org/anthology/N18-2009. https://doi.org/10.18653/v1/N18-2009
11. W.-T. Hsu, C.-K. Lin, M.-Y. Lee, K. Min, J. Tang, M. Sun, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. A unified model for extractive and abstractive summarization using inconsistency loss (Association for Computational Linguistics, Melbourne, Australia, 2018), pp. 132–141. https://www.aclweb.org/anthology/P18-1013. https://doi.org/10.18653/v1/P18-1013
12. S. Chopra, M. Auli, A. M. Rush, in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Abstractive sentence summarization with attentive recurrent neural networks (Association for Computational Linguistics, San Diego, California, 2016), pp. 93–98. https://www.aclweb.org/anthology/N16-1012. https://doi.org/10.18653/v1/N16-1012
13. D. He, H. Lu, Y. Xia, T. Qin, L. Wang, T.-Y. Liu, in *Advances in Neural Information Processing Systems 30*, ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Decoding with value networks for neural machine translation (Curran Associates, Inc., 2017), pp. 178–187. http://papers.nips.cc/paper/6622-decoding-with-value-networks-for-neural-machine-translation.pdf
14. B. Dzmitry, C. Kyunghyun, B. Yoshua, Neural machine translation by jointly learning to align and translate. Comput. Sci. (2014). 1409.0473v6
15. J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res. **12**(Jul), 2121–2159 (2011)
16. C. Y. Lin, E. Hovy, in *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. Automatic evaluation of summaries using

n-gram co-occurrence statistics, (2003), pp. 150–157. https://www.aclweb.org/anthology/N03-1020

17. M. D. Zeiler, ADADELTA: an adaptive learning rate method. CoRR. **abs/1212.5701** (2012). 1212.5701
18. M. Zhanyu, L. Yuping, K. W. Bastiaan, W. Liang, G. Jun, Variational Bayesian learning for Dirichlet process mixture of inverted Dirichlet distributions in non-Gaussian image feature modeling. IEEE Trans. Neural Netw. Learn. Syst. **30**(2), 449–463 (2019). https://doi.org/10.1109/TNNLS.2018.2844399
19. M. Zhanyu, Y. Hong, C. Wei, G. Juo, Short utterance based speech language identification in intelligent vehicles with time-scale modifications and deep bottleneck features. IEEE Trans. Veh. Technol. **68**(1), 121–128 (2019). https://doi.org/10.1109/TVT.2018.2879361
20. K. Zhang, N. Liu, X. Yuan, X. Guo, C. Gao, Z. Zhao, Fine-grained age estimation in the wild with attention LSTM networks. IEEE Trans. Circ. Syst. Video Technol. (TCSVT) (2019). Accepted. https://doi.org/10.1109/tcsvt.2019.2936410
21. L. Xiaoxu, Y. Liyun, C. Dongliang, M. Zhanyu, C. Jie, Dual cross-entropy loss for small-sample fine-grained vehicle classification. IEEE Trans. Veh. Technol. (TVT). **68**(5), 4204–4212 (2019)
22. M. Zhanyu, X. Jing-Hao, L. Arne, T. Zheng-Hua, Y. Zhen, G. Jun, Decorrelation of neutral vector variables: theory and applications. IEEE Trans. Neural Netw. Learn. Syst. **29**(1), 129–143 (2018). https://doi.org/10.1109/TNNLS.2016.2616445
23. M. Zhanyu, C. Dongliang, X. Jiyang, D. Yifeng, W. Shaoguo, L. Xiaoxu, S. Zhongwei, G. Jun, Fine-grained vehicle classification with channel max pooling modified CNNs. IEEE Trans. Veh. Technol. **68**(4), 3224–3233 (2019)
24. M. Zhanyu, X. Jiyang, L. Yuping, T. Jalil, X. Jing-Hao, G. Jun, Insights into multiple/single lower bound approximation for extended variational inference in non-Gaussian structured data modeling. IEEE Trans. Neural Netw. Learn. Syst., 1–15 (2019). https://doi.org/10.1109/TNNLS.2019.2899613
25. K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, P. Blunsom, in *Advances in Neural Information Processing Systems 28*, ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Teaching machines to read and comprehend (Curran Associates, Inc., 2015), pp. 1693–1701. http://papers.nips.cc/paper/5945-teaching-machines-to-read-and-comprehend.pdf

Publisher's Note