Keywords-Guided Abstractive Sentence Summarization

Haoran Li[1], **Junnan Zhu**[2,3], **Jiajun Zhang**[2,3], **Chengqing Zong**[2,3,4], **Xiaodong He**[1]

[1] JD AI Research
[2]National Laboratory of Pattern Recognition, Institute of Automation, CAS
[3] University of Chinese Academy of Sciences
[4] CAS Center for Excellence in Brain Science and Intelligence Technology
{lihaoran24, xiaodong.he}@jd.com
{junnan.zhu, jjzhang, cqzong}@nlpr.ia.ac.cn

Abstract

We study the problem of generating a summary for a given sentence. Existing researches on abstractive sentence summarization ignore that keywords in the input sentence provide significant clues for valuable content, and humans tend to write summaries covering these keywords. In this paper, we propose an abstractive sentence summarization method by applying guidance signals of keywords to both the encoder and the decoder in the sequence-to-sequence model. A multi-task learning framework is adopted to jointly learn to extract keywords and generate a summary for the input sentence. We apply keywords-guided selective encoding strategies to filter source information by investigating the interactions between the input sentence and the keywords. We extend pointer-generator network by a dual-attention and a dual-copy mechanism, which can integrate the semantics of the input sentence and the keywords, and copy words from both the input sentence and the keywords. We demonstrate that multi-task learning and keywords-oriented guidance facilitate sentence summarization task, achieving better performance than the competitive models on the English Gigaword sentence summarization dataset.

Introduction

Sentence summarization is a task that creates a condensed version of a long sentence[1]. Different from extractive methods (Cheng and Lapata 2016; Jadhav and Rajan 2018; Dong et al. 2018; Zhang et al. 2018), which select a subset of text units in the original text to form the summary, abstractive methods (Rush, Chopra, and Weston 2015; Takase et al. 2016; Chen et al. 2016; See, Liu, and Manning 2017; Tan, Wan, and Xiao 2017; Zhou et al. 2017; Narayan, Cohen, and Lapata 2018; Lebanoff, Song, and Liu 2018; Zhu et al. 2019) can generate novel words not present in the input. Compared with extractive methods, abstractive summarization is much closer to the way human make a summary, while it is more challenging. Intuitively, some important words (a.k.a. keywords) in the original sentence provide significant clues for the main points about the sentence. Humans tend to write summaries containing these keywords and then perform necessary modifications to ensure the fluency and grammatically correctness. Thus, we believe that it will be easier for a machine to generate a summary with the help of keywords. Existing work has not explored the effectiveness of keywords for sentence summarization task, and our work focuses on it.

[1]For sentence compression task (Clarke 2008), the word of the output must be in the input, while the vocabulary for sentence summarization is not constrained.

Observation:
Input sentence: France and Germany called on world leaders Monday to take rapid action to press for the closure of Ukraine 's Chernobyl nuclear plant , site of the world 's worst ever nuclear disaster .
Reference summary: World leaders urged to back Chernobyl closure plan .

Solution:
Step1. Extracting keywords: world leaders closure Chernobyl
Step2. Generating summary guided by the keywords: World leaders called for action on Chernobyl closure .

Figure 1: The overlapping keywords (marked in red) between the input sentence and the reference summary cover the main ideas of the input sentence. Our motivation is to generate summary guided by the keywords extracted from the input sentence.

Given a pair of input sentence and reference summary, we can roughly take the overlapping words (except for stopwords) as the keywords. As shown in Figure 1, the overlapping words cover the gist of the input. For example, the keywords "*closure*" and "*Chernobyl*" can guide us to focus on the part "*closure of Ukraine 's Chernobyl nuclear plant*" of the original sentence, which is highly related to "*Chernobyl closure plan*" in the summary. This phenomenon is common for sentence summarization. In the Gigaword sentence summarization dataset (Rush, Chopra, and Weston 2015), over half of the words in the summary are presented in the source sentence.

Existing researches show that keywords are beneficial for extractive summarization (Saggion and Lapalme 2002; Zhang, Zincir-Heywood, and Milios 2004; Wang and Cardie

2013). They point out that keywords compose the main body of the sentence, which are regarded as the indicators for important sentence selection. On the other hand, keywords are proved useful for decoding process in abstractive document-level summarization task (Li et al. 2018a; Gehrmann, Deng, and Rush 2018). We argue that keywords can point out valuable content in the input sentence, which can guide the summarizer to capture the gist of the input in the process of both encoding and decoding.

A prerequisite for our model is the keywords of the input sentence. For training, we can directly use the overlapping words between the input sentence and the reference summary as the ground-truth keywords, while the ground-truth keywords are not available for testing. Consequently, a keyword extractor is required. Sentence summarization and keyword extraction both aim to mine the primary ideas of the input text but with different forms of output. Sentence summarization aims to express the main meanings of the input with a complete sentence, while keyword extraction is to select the important words from the input. Thus, these two tasks both require the capacity of the encoder to recognize the crucial text fragments in the source sentence. Based on this, we adopt a multi-task learning framework to model sentence summarization and keyword extraction jointly, which is expected to be beneficial for both tasks. Then we explore the effectiveness of the keywords for sentence summarization task through following three strategies. We first apply keywords-guided selective encoding strategies to filter source information, and then we dynamically integrate the semantics of the input sentence and the keywords to build context representation via dual-attention. Furthermore, we extend the copy mechanism to a dual-copy mode that can copy words from both the input sentence and the keywords. The framework of our model is shown in Figure 2.

Our main contributions are as follows:

- We propose an abstractive sentence summarization method guided by the keywords in the original sentence.
- Our encoder builds an optimized latent representation by keywords-guided selective encoding.
- Our decoder dynamically combines the information of the original sentence and the keywords via dual-attention, and we propose a dual-copy mechanism which facilitates copying words from both the input sentence and the keywords.
- We achieve significantly better performance than the competitive methods on the English Gigaword dataset.

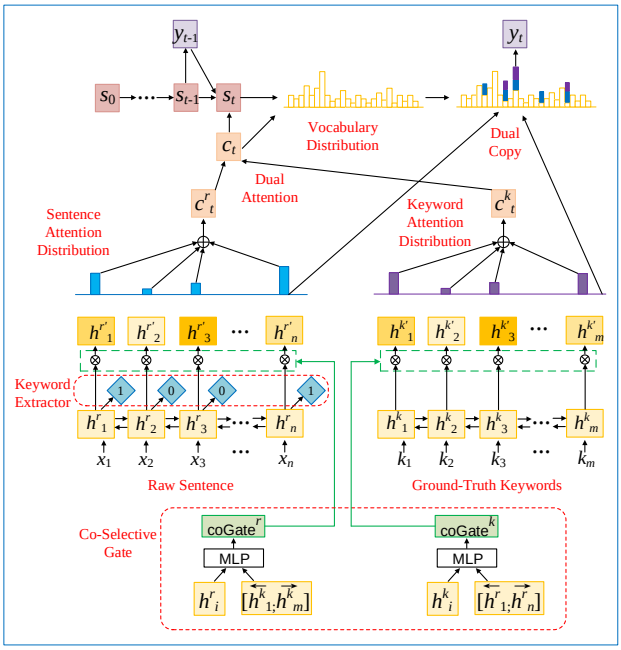

Figure 2: The framework of our model with co-selective encoding. During training, a BiLSTM reads the original sentence $(x_1, x_2, \cdots, x_n)$ and the ground-truth keywords $(k_1, k_2, \cdots, k_m)$ into the first-level hidden states h_i^r and h_i^k. A jointly trained keyword extractor takes h_i^r as the input to predict whether the input word is a keyword or not. Co-selective encoding layer builds the second-level hidden states $h_i^{r'}$ and $h_i^{k'}$. Then the summary is generated via dual-attention and dual-copy for both the original sentence and the keyword sequence. During testing, the ground-truth keywords are replaced by the keywords predicted by our trained keyword extractor.

Related Work

Abstractive Text Summarization

Seq2seq model is the dominating framework for abstractive text summarization. Rush, Chopra, and Weston (2015) are the first to apply the seq2seq model to abstractive sentence summarization. They propose an attentive CNN encoder and a neural network language model (Bengio et al. 2003) decoder. Chopra, Auli, and Rush (2016) and Nallapati et al. (2016) further extend the RNN-based summarization model. Gu et al. (2016), Zeng et al. (2016) and Gulcehre et al. (2016) introduce a copy mechanism into seq2seq learning. See, Liu, and Manning (2017) incorporate the pointer-generator model with the coverage mechanism. Zhou et al. (2017) employ a selective encoding mechanism to filter secondary information. Tan, Wan, and Xiao (2017) propose a graph-based attention to tackle document-level summarization. Cao et al. (2018) and Li et al. (2018b) solve the problem of fake facts in a summary.

Keywords-Guided Summarization

Keywords have been proved beneficial for extractive text summarization systems. Zhang, Zincir-Heywood, and Milios (2004) extract the most significant sentences based on the density of keywords. Wang and Cardie (2013) train a classifier to identify summary-worthy phrases and then apply a heuristic strategy to rank the sentences for the abstract. Wan, Yang, and Xiao (2007) attempt to use a reinforcement approach to extract keywords and summarize simultaneously, supposing that important sentences usually contain keywords and keywords are also usually seen in important sentences. Recently, keywords have been used in abstractive

document-level summarization task. Li et al. (2018a) use the keywords to calculate attention distribution and copy probability. Gehrmann, Deng, and Rush (2018) propose a content selector to restrict the summarization model to copy phrases from the source document. Above-mentioned work focuses on enhancing the decoder. We believe that keywords can eliminate the redundant information in the source, and thus, we propose keywords-guided selective mechanisms to improve the source encoding representations. Besides, our decoder can dynamically combine the information of the input sentence and the keywords to generate summaries.

Our Proposed Model

Overview

The input of sentence summarization task is a long sentence, and the output is a condensed summary. Our hypothesis for this task is that the keywords can provide essential clues for the gist of the input sentence. The standard seq2seq model takes into account of all source words, and attention mechanism may work as a soft keyword extractor. We propose a novel pointer-generator-based abstractive sentence summarization method incorporating a keyword extractor, which can explicitly point out the valuable content of the input sentence.

A prerequisite for our model is the keywords for the input sentence. For training, we take the words appearing both in the input sentence and the reference summary as the ground-truth keywords. To acquire the keywords for testing, we train a keyword extractor by multi-task learning (MTL) with summarization model. The encoder read the input sentence into a latent representation sequence, and a softmax layer over latent representation predicts whether a word is a keyword or not. Our summarization model consists of three major parts: an encoder with selective encoding strategy, a generator combining context information of the original sentence and the keywords via dual-attention, and a pointer copying words from both the original sentence and the keywords via dual-copy.

Here are the **main steps** of our model.

- **Step 1.** Extracting overlapping words between the input and the reference as the ground-truth keywords.
- **Step 2.** Multi-task learning: generating summary using the input sentence and the ground-truth keywords; training the keyword extractor.
- **Step 3.** Generating keywords using the trained keywords extractor for the input sentence in the training set and then fine-tuning the sentence summarizer using the original sentence and the predicted keywords.
- **Step 4.** During testing, first generating keywords using the trained keywords extractor for the input sentence and then producing the summary using the input sentence and the predicted keywords.

Ground-Truth Keyword Generation

Standard Gigaword dataset for sentence summarization task does not provide the keywords for the input sentence. To train our keyword extractor and keywords-guided summarizer, we roughly regard the overlapping words (stop-words are excluded) between the input sentence and the reference summary as the ground-truth keywords (sentence-summary pair in the Gigaword dataset is bound to have overlapping non-stop-words).

Shared Text Encoder

The BiLSTM encodes input text forwardly and backwardly to generate two sequences of the hidden states: $(\overrightarrow{h}_1, \cdots, \overrightarrow{h}_n)$ and $(\overleftarrow{h}_1, \cdots, \overleftarrow{h}_n)$, respectively, where:

$$\overrightarrow{h}_i = \mathrm{LSTM}(\mathrm{E}[\mathrm{x_i}], \overrightarrow{h}_{\mathrm{i-1}}) \quad (1)$$

$$\overleftarrow{h}_i = \mathrm{LSTM}(\mathrm{E}[\mathrm{x_i}], \overleftarrow{h}_{\mathrm{i+1}}) \quad (2)$$

$E[x_i]$ is the embedding for word x_i. The final hidden representation h_i is the concatenation of the forward and backward vectors: $h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i]$.

Sentence summarization task and keyword extraction task are very similar in the sense that both aim to select important information contained in the input sentence. The output of sentence summarization is a complete sentence, while the output for keyword extraction is a set of words. It requires different modules to generate corresponding output for various tasks, while for the encoders, we believe they can benefit from sharing parameters to promote the capacity of capturing the gist of the input text. To this end, we use a shared text encoder to generate hidden state sequences for both the original sentence and the keyword sequence following Equation 1 and 2. Hereafter, h_i^r and h_i^k denote the hidden representations for the input sentence and the keywords, respectively.

Keyword Extraction

For keyword extraction task, the output layer is a softmax classifier over the hidden representation h_i^r for each word in the sentence. The classifier predicts one of the following two labels: '1' for the keywords, and '0' for the non-keywords.

We also try BiLSTM-CRF model (Huang, Xu, and Yu 2015) which extends the BiLSTM model with a CRF layer allowing the model to use sequence-level tag information for sequence prediction, but our experimental results show that BiLSTM-CRF model leads to a similar performance (0.2% higher accuracy) with around 20% more trainable parameters and doubled training time compared with BiLSTM-softmax model.

Keywords-Guided Selective Encoding

We are inspired by the work of Zhou et al. (2017) on a new paradigm to encode sentences. Specifically, they propose a selective encoding mechanism to model the selection process for sentence summarization. A selective gate is applied to h_i^r to construct a second-level tailored representation $h_i^{r'}$.

The selective gate in their model is expected to maintain the important information from an encoded sentence by exploring the semantic relationship between every word in the input sentence and the whole sentence. This kind of selective encoding strategy can be referred to as self-selective encoding because the selective signal comes from the original

sentence itself. Compared with the original sentence, keywords contain more condensed semantics, and thus, the selective signal could be more powerful. Unlike self-selective, we leverage information of the keywords to construct the selective gate. We propose two keywords-guided selective encoding strategies: keywords-selective which builds $h_i^{r'}$ for the sentence guided by the keywords and co-selective encoding which builds $h_i^{r'}$ and $h_i^{k'}$ for the input sentence and the keywords, respectively. More details are as follows.

Self-Selective Encoding A self-selective gate vector for each h_i^r is computed as follows:

$$selfGate_i^s = \sigma(\mathbf{W}_r h_i^r + \mathbf{U}_r a^r) \tag{3}$$

where σ denotes the sigmoid function, and $a^r = [\overrightarrow{h_n^r}; \overleftarrow{h_1^r}]$ is the sentence representation. Then, $h_i^{r'}$ is computed as follows:

$$h_i^{r'} = h_i^r \odot selfGate_i^s \tag{4}$$

where $\odot$ is element-wise multiplication.

Keywords-Selective Encoding Keywords-selective gate uses keywords to guide the encoding of the original sentence as follows:

$$keyGate_i^r = \sigma(\mathbf{W}_k h_i^r + \mathbf{U}_k a^k) \tag{5}$$

where $a^k = [\overrightarrow{h_n^k}; \overleftarrow{h_1^k}]$ is the keyword sequence representation. Then, $h_i^{r'}$ is computed as follows:

$$h_i^{r'} = h_i^r \odot keyGate_i^r \tag{6}$$

Co-Selective Encoding Beyond selecting encoding for the sentence, we argue that each keyword contributes differently to the summarization task, and thus, we propose a co-selective encoding to select information for both the sentence and the keywords jointly.

$$coGate_i^r = \sigma(\mathbf{W}_p h_i^r + \mathbf{U}_p a^k) \tag{7}$$

$$coGate_i^k = \sigma(\mathbf{W}_q h_i^k + \mathbf{U}_q a^r) \tag{8}$$

Then, $h_i^{r'}$ and $h_i^{k'}$ are computed as follows:

$$h_i^{r'} = h_i^r \odot coGate_i^r \tag{9}$$

$$h_i^{k'} = h_i^k \odot coGate_i^k \tag{10}$$

Dual-Attention Generator

Pointer-generator network is a seq2seq model with copy mechanism, which predicts words based on probability distributions from the generator and the pointer (Vinyals, Fortunato, and Jaitly 2015). The generator applies a dual-attention mechanism to generate the context vector based on attention over both the source sentence and the extracted keywords.

We use an LSTM with attention as the decoder to produce the output summary. At each timestep t, LSTM reads the previous decoder state s_{t-1}, predicted output y_{t-1}, and context vector c_{t-1} to compute the current decoder state s_t as follows:

$$s_t = \text{LSTM}(s_{t-1}, y_{t-1}, c_t) \tag{11}$$

The decoder's hidden state s_0 is initialized as follows:

$$s_0 = \tanh(\mathbf{W}_h[\overrightarrow{h_n^{r'}}; \overleftarrow{h_1^{r'}}]) \tag{12}$$

The input for our model includes the original sentence and the keywords, and thus we first construct the context vector for the sentence and the keywords with the attention mechanism. Then we obtain the final context vector c_t.

We compute the sentence context vector c_t^r as follows:

$$c_t^r = \sum_{i=1}^{N} \alpha_{t,i}^r h_i^{r'} \tag{13}$$

where each vector is weighted by the attention weight $\alpha_{t,i}^r$, as calculated in Equations 14 and 15 as follows:

$$e_{t,i}^r = v_a^T \tanh(\mathbf{U}_a s_{t-1} + \mathbf{W}_a h_i^{r'}) \tag{14}$$

$$\alpha_{t,i}^r = \frac{\exp(e_{t,i}^r)}{\sum_{j=1}^{N} \exp(e_{t,j}^r)} \tag{15}$$

Similarly, the keyword attention $\alpha_{t,i}^k$ and keyword context vector c_t^k can be calculated using $h_i^{k'}$ and s_{t-1}. Next, we apply three approaches to fuse c_t^r and c_t^k into the final context vector c_t, including concatenation fusion, gated fusion, and hierarchical fusion.

Concatenation Fusion This fusion method simply concatenates two context vectors:

$$c_t = [c_t^r; c_t^k] \tag{16}$$

Gated Fusion We first compute a fusion gate vector using two context vectors and then combine context vectors by the gate:

$$g_t = \sigma(\mathbf{W}_g c_t^r + \mathbf{U}_g c_t^k) \tag{17}$$

$$c_t = g_t \odot c_t^r + (1 - g_t) \odot c_t^k \tag{18}$$

Hierarchical Fusion Gated fusion combines the two context vectors with the gate related to the interaction between them. Intuitively, the fusion process should reflect the relative informativeness of the sentence and the keywords toward the decoder state s_{t-1}. We adopt a hierarchical attention mechanism which aims to establish a target-oriented bond between the original sentence and the keywords as follows:

$$\beta_t^r = \sigma(\mathbf{U}_\beta s_{t-1} + \mathbf{W}_\beta c_t^r) \tag{19}$$

$$\beta_t^k = \sigma(\mathbf{U}_\beta s_{t-1} + \mathbf{W}_\beta c_t^k) \tag{20}$$

$$c_t = \beta_t^r c_t^r + \beta_t^k c_t^k \tag{21}$$

where β_t^r and β_t^k are two scalars.

Then, we can calculate probability distribution P_{gen} over all words in the target vocabulary is calculated as follows:

$$P_{gen}(w) = \text{softmax}(\mathbf{W}_h s_t + \mathbf{V}_h c_t + b_h) \tag{22}$$

Dual-Copy Pointer

A general pointer copies a word w from the source via pointing:

$$P_{copy}(w) = \sum\nolimits_{i:x_i=w} \alpha_{t,i} \tag{23}$$

We propose a dual-copy pointer that copies a word w from both the source sequence and keyword sequence as follows:

$$P_{copys}(w) = \sum\nolimits_{i:x_i=w} \alpha^r_{t,i} \tag{24}$$

$$P_{copyk}(w) = \sum\nolimits_{i:k_i=w} \alpha^k_{t,i} \tag{25}$$

$$\tag{26}$$

The final distribution is a weighted sum of the generation distribution and dual-copy distribution:

$$\lambda_t = \text{sigmoid}(w_a^T c_t + u_g^T s_t + v_g^T y_{t-1} + b_g) \tag{27}$$

$$P(w) = \lambda_t P_{gen}(w) + \frac{1}{2}(1-\lambda_t)(P_{copys}(w) + P_{copyk}(w)) \tag{28}$$

The loss function $\mathcal{L}_t$ for each timestep t is the negative log-likelihood of the ground-truth target word y_t :

$$\mathcal{L}_t = -\log P(y_t) \tag{29}$$

Multi-Task Learning

In our MTL setup, summary generation task is much more complicated than keyword extraction, leading to different learning difficulties and convergence rates. Therefore, summary generation is regarded as the central task and keyword extraction as the auxiliary task. We optimize the two tasks alternatively during training until convergence. Let γ be the number of mini-batches of training for keyword extracting after 100 mini-batches of training for sentence summarization (Pasunuru, Guo, and Bansal 2017). We adopt $\gamma = 10$ in our experiments.

Fine-Tuning with Generated Keywords

Our model is trained with the ground-truth keywords, instead of using the results of our keyword detector, while the ground-truth keywords cannot be obtained during testing. Using ground-truth keywords causes it to converge faster, but the mismatching between the processes of training and testing may exhibit instability. To solve this problem, we propose to fine-tune our model with the predicted keywords generated by our keyword detector as the input after the training process converges using ground-truth keywords.

Experiment

Dataset

We conduct experiments on the English Gigaword dataset, which has about 3.8 million training sentence-summary pairs. We use 8, 000 pairs as the validation set and 2, 000 pairs as the test set, provided by Zhou et al. (2017).

Implementation Details

We set the size of word embedding and LSTM hidden state to 300 and 512, respectively. Adam optimizer is applied with the learning rate of 0.0005, momentum parameters $\beta_1 = 0.9$ and $\beta_1 = 0.999$, and $\epsilon = 10^{-8}$. We use dropout (Srivastava et al. 2014) with probability of 0.2 and gradient clipping (Pascanu, Mikolov, and Bengio 2013) with range $[-1, 1]$. The mini-batch size is set to 64. At training time, we test ROUGE-2 (Lin 2004) F1 score on the validation set for every 2,000 batches, and we halve the learning rate if model performance drops. We use the halved learning rate for fine-tuning. At test time, we use beam search with a beam size of 10 to generate the summary. We report ROUGE F1 scores.

Comparative Methods

ABS. Rush, Chopra, and Weston (2015) use an attentive CNN encoder and neural network language model decoder to summarize a sentence.

SEASS. Zhou et al. (2017) present a selective encoding model to control the information flow from the encoder to the decoder.

PG. See, Liu, and Manning (2017) introduce a hybrid pointer-generator model that can copy words from the source sentence via pointing.

KIGN. Li et al. (2018a) adopt key information to guide the summarization generation. They use the key information representation as the extra input to calculate the attention distribution and the copy probability in the pointer mechanism.

Bottom-up. Gehrmann, Deng, and Rush (2018) employ a content selection model to identify important phrases in the input document.

Our models. We first take a seq2seq (S2S) model (Bahdanau, Cho, and Bengio 2015) as the baseline model with different input: the original sentence only (**S2S-Sentence**) , the keywords only (**S2S-Keywords**), a new sentence merging the original sentence and keywords (**S2S-Sentence&Keywords**). We compare different selective-based models, including **Self-Selective**, **Keywords-Selective**, and **Co-Selective**. We also report the results for three approaches to fuse the context of the original sentence and keywords, including **Concatenation** fusion, **Gated** fusion, and **Hierarchical** fusion. Finally, we conduct experiment with **PG** and our proposed dual-copy pointer-generator (**DualPG**).

Main Results

Table 1 shows that our proposed models perform better than the models without keyword guidance. Among our models with different selective mechanisms, the experimental results are improved steadily as more keyword guidance signals added into the models, from **Self-selective** to **Co-Selective**. The models with **Hierarchical** fusion exhibit advantages over those with other fusions. The model with **Co-Selective** encoding, **Hierarchical** fusion decoding, and **DualCopy** obtains the highest ROUGE score, which outperforms **S2S-Sentence** absolute 2.05% ROUGE-1 score,

Table 1: Main results (%). *Concat*, *Gated*, and *Hier* denote *Concatenation*, *Gated*, and *Hierarchical Fusion*, respectively. Our *Co-Selective+Hier+DualPG* model performs significantly better than other baselines by the 95% confidence interval in the ROUGE script.

Method		R-1	R-2	R-L
ABS		37.41	15.87	34.70
SEASS		46.86	24.58	43.53
PG		46.97	24.63	43.66
KIGN		46.18	23.93	43.44
Bottom-up		45.80	23.61	42.54
S2S-Sentence		45.09	23.58	42.37
S2S-Keywords		44.85	20.54	41.61
S2S-Sentence&Keywords		46.14	24.07	43.30
Self-Selective	Concat	46.23	24.13	43.26
	Gated	46.44	24.31	43.44
	Hier	46.51	24.32	43.45
Keywords-Selective	Concat	46.32	24.11	43.38
	Gated	46.70	24.48	43.59
	Hier	46.72	24.50	43.81
Co-Selective	Concat	46.53	24.15	43.24
	Gated	46.71	24.53	43.63
	Hier	46.80	24.75	43.83
Co-Selective	Concat+PG	46.68	24.33	43.31
	Gated+PG	46.91	24.61	43.71
	Hier+PG	46.93	24.83	43.92
Co-Selective	Concat+DualPG	47.05	24.39	43.77
	Gated+DualPG	47.13	24.87	44.34
	Hier+DualPG	**47.14**	**25.06**	**44.39**

1.48% ROUGE-2 score, and 2.02% ROUGE-L score. **S2S-Keywords** with only the keywords as the input degrades the performance, showing that missing information from the keywords is also necessary.

SEASS brings in an excellent promotion for sentence summarization task with the help of implicit emphases on the keywords, and our best model achieves a better performance with explicit use of the keywords. **KIGN** and **Bottom-up** are keywords-based document summarization models that mainly focus on improving the decoder, and we apply them to sentence summarization task. Similar to **S2S-Keywords**, **Bottom-up** also blocks out the words detected as non-keywords with high probabilities, which also leads to unsatisfactory results. **KIGN** is proposed to use the keywords selected from the input document using unsupervised TextRank algorithm (Mihalcea and Tarau 2004) based on words co-occurrence relation, which may not be appropriate for sentence-level keyword extraction (Table 4 shows that keyword extraction performance for TextRank is quite poor compared with our model). Thus, we further implement **KIGN** with keywords extracted by our model, and the results are: 46.84% for ROUGE-1, 24.33% for ROUGE-2, and 43.98% for ROUGE-L, which are better than those of the original **KIGN** model due to mitigating negative impact of keyword extraction errors, while this model still performs worse than ours. The comparison with existing

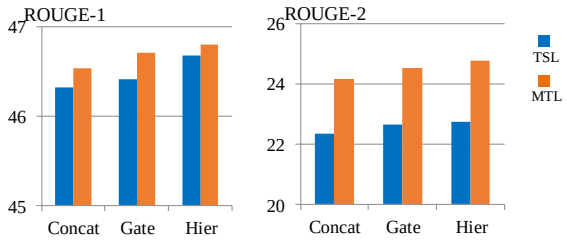

Figure 3: Results of Co-Selective models with MTL and two-stage learning (TSL) for summarization task.

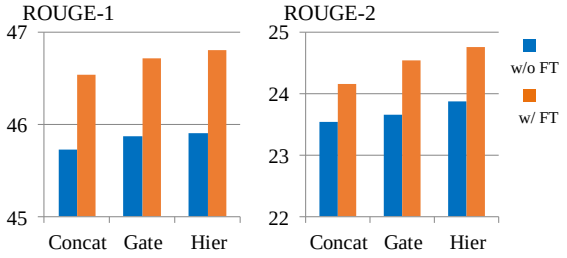

Figure 4: Results of Co-Selective models with (w/) and without (w/o) fine-tuning (FT) for summarization task.

keywords-based models demonstrates the effectiveness of the keywords to enhance both the encoder and the decoder.

Analysis and Discussion

To get better insights into our model, we conduct further analysis on (1) upper bound performance, (2) multi-task learning, (3) fine-tuning, and (4) selective encoding mechanism.

Upper Bound Performance

We explore the upper bound performance for our keyword-guided sentence summarization model. We do this by directly using the ground-truth keywords for both training and testing. In this way, the keyword extraction error is eliminated. The results are shown in Table 2. With this oracle setting, ROUGE score improvements are more than 20% over seq2seq model. Although the improvement is impressive, it is based on the premise that we know which words should

Table 2: Upper bound performances with the ground-truth keywords for both training and testing.

Method		R-1	R-2	R-L
S2S-Sentence (baseline)		45.09	23.58	42.37
S2S-Keywords		65.00	37.29	58.96
S2S-Sentence&Keywords		66.35	41.34	61.63
KIGN		66.00	41.19	60.66
Bottom-up		65.11	38.12	59.39
Co-Selective	Concat+DualPG	66.13	41.55	60.47
	Gated+DualPG	67.35	42.82	61.85
	Hier+DualPG	67.61	42.94	62.07

Table 3: Heat maps for our model with co-selective and self-selective encoding.

Self-selective gate for input sentence	about #,### people gathered saturday in a central prague park to support the legalization of marijuana
Summary generated by self-selective model	about #,### people gathered in central prague
Co-selective gate for input sentence	about #,### people gathered saturday in a central prague park to support the legalization of marijuana
Co-selective gate for keywords	people prague support legalization marijuana
Summary generated by co-selective model	people gather to support legalization of marijuana
Reference summary	thousand gather to support legalization of marijuana

Table 4: Comparison of MTL and two-stage learning for keyword extraction. Accuracy is for all the words, and F1 score is for the keywords.

Method		Accuracy	F1
TextRank		73.25	31.45
Two-Stage Learning		84.37	59.35
MTL with Co-selective Encoding	Concat	85.34	60.11
	Gate	85.75	60.24
	Hier	85.81	60.27

Table 5: Manual evaluation.

Method	Readability	Informativeness
S2S-Sentence	3.02	3.16
Co-Select+Hier	3.36	3.47
Co-Select+Hier+DualPG	3.61	3.79
Reference	4.17	4.54

be included in the summary in advance. Actually, choosing a golden set of keywords may be difficult even for humans. We believe this experiment holds out a promising prospect for further development of sentence summarization task.

MTL v.s. Two-Stage Learning

MTL involves sharing parameters between related tasks, whereby each task can benefit from extra information on other tasks in the training process. We compare the performance for MTL and two-stage learning (TSL, learning two tasks independently). Figure 3 shows the results for sentence summarization, and Table 4 shows the results for keyword extraction. Our models with MTL achieve better performances than those with TSL, proving that summarization and keyword extraction can benefit from each other. In addition, we also train our summarizer by TSL directly with the predicted keywords as input (without pre-training with the ground-truth keywords), and it drops about 1% ROUGE-2 score.

Effectiveness of Fine-tuning

In this section, we evaluate the Co-Selective summarization models with and without fine-tuning. The results in Figure 4 show that fine-tuning steadily enhances the performance, improving absolute ROUGE-2 score about 1% for **Hierarchical** fusion.

Manual Evaluation

We employ six graduate students to evaluate readability and informativeness of summaries generated by different methods with a score from 1 (worst) to 5 (best). Each participant is provided 100 sentences randomly sampled from the test set. As shown in Table 5, our method with keyword guidance achieves higher scores than other methods except for the reference.

Selective Encoding Visualization

Following Zhou et al. (2017), we visualize the selective gate values with salience heat maps shown in Table 3. For our model with co-selective gate, the important words are selected correctly by the aid of keywords, while the output of the model with self-selective gate is mismatched with the reference summary because of inaccurate selective values.

Conclusion

This paper addresses the sentence summarization task, namely, how to transform a sentence into a short-length summary. Our proposed model can take advantage of keywords achieving better performance than the competitive methods. We adopt a multi-task learning framework to extract keywords and generate summaries jointly. We design keywords-guided selective encoding strategies to select important information during encoding. We adopt a dual-attention structure to dynamically integrate context information of the input sentence and the keywords. We propose a dual-copy mechanism to copy the words from the input sentence and the keywords. Experimental results on standard dataset verify the effectiveness of keywords for sentence summarization task. Oracle testing with ground-truth keywords leads to absolute 20% ROUGE-2 score improvement over baseline, indicating a promising future direction based on keyword extraction for sentence summarization task.

Acknowledgments

The research work descried in this paper has been supported by the National Key Research and Development Program of China under Grant No. 2017YFC0820700.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Bengio, Y.; Ducharme, R.; Vincent, P.; and Janvin, C. 2003. A neural probabilistic language model. *JMLR* 3:1137–1155.

Cao, Z.; Wei, F.; Li, W.; and Li, S. 2018. Faithful to the original: Fact aware neural abstractive summarization. In *Proceedings of AAAI*.

Chen, Q.; Zhu, X.; Ling, Z.; Wei, S.; and Jiang, H. 2016. Distraction-based neural networks for modeling documents. In *Proceedings of IJCAI*, 2754–2760.

Cheng, J., and Lapata, M. 2016. Neural summarization by extracting sentences and words. In *Proceedings of ACL*.

Chopra, S.; Auli, M.; and Rush, A. M. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of NAACL*, 93–98.

Clarke, J. 2008. *Global inference for sentence compression : an integer linear programming approach*. Ph.D. Dissertation, University of Edinburgh, UK.

Dong, Y.; Shen, Y.; Crawford, E.; van Hoof, H.; and Cheung, J. C. K. 2018. Banditsum: Extractive summarization as a contextual bandit. In *Proceedings of EMNLP*, 3739–3748.

Gehrmann, S.; Deng, Y.; and Rush, A. 2018. Bottom-up abstractive summarization. In *Proceedings of EMNLP*.

Gu, J.; Lu, Z.; Li, H.; and Li, V. O. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of ACL*, 1631–1640.

Gulcehre, C.; Ahn, S.; Nallapati, R.; Zhou, B.; and Bengio, Y. 2016. Pointing the unknown words. In *Proceedings of ACL*, 140–149.

Huang, Z.; Xu, W.; and Yu, K. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv:1508.01991*.

Jadhav, A., and Rajan, V. 2018. Extractive summarization with SWAP-NET: Sentences and words from alternating pointer networks. In *Proceedings of ACL*.

Lebanoff, L.; Song, K.; and Liu, F. 2018. Adapting the neural encoder-decoder framework from single to multi-document summarization. In *Proceedings of EMNLP*.

Li, C.; Xu, W.; Li, S.; and Gao, S. 2018a. Guiding generation for abstractive text summarization based on key information guide network. In *Proceedings of NAACL*.

Li, H.; Zhu, J.; Zhang, J.; and Zong, C. 2018b. Ensure the correctness of the summary: Incorporate entailment knowledge into abstractive sentence summarization. In *Proceedings of COLING*, 1430–1441.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*.

Mihalcea, R., and Tarau, P. 2004. Textrank: Bringing order into text. In *Proceedings of EMNLP*.

Nallapati, R.; Zhou, B.; dos Santos, C.; Gulcehre, C.; and Xiang, B. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of CoNLL*, 280–290.

Narayan, S.; Cohen, S. B.; and Lapata, M. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of EMNLP*.

Pascanu, R.; Mikolov, T.; and Bengio, Y. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of ICML*, 1310–1318.

Pasunuru, R.; Guo, H.; and Bansal, M. 2017. Towards improving abstractive summarization via entailment generation. In *Proceedings of the Workshop on New Frontiers in Summarization*.

Rush, A. M.; Chopra, S.; and Weston, J. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of EMNLP*, 379–389.

Saggion, H., and Lapalme, G. 2002. Generating indicative-informative summaries with SumUM. *Computational Linguistics* 28(4):497–526.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of ACL*, 1073–1083.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*.

Takase, S.; Suzuki, J.; Okazaki, N.; Hirao, T.; and Nagata, M. 2016. Neural headline generation on abstract meaning representation. In *Proceedings of EMNLP*.

Tan, J.; Wan, X.; and Xiao, J. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of ACL*, 1171–1181.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. In *NeurIPS*, 2692–2700.

Wan, X.; Yang, J.; and Xiao, J. 2007. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Proceedings of ACL*.

Wang, L., and Cardie, C. 2013. Domain-independent abstract generation for focused meeting summarization. In *Proceedings of ACL*.

Zeng, W.; Luo, W.; Fidler, S.; and Urtasun, R. 2016. Efficient summarization with read-again and copy mechanism. *arXiv:1611.03382*.

Zhang, X.; Lapata, M.; Wei, F.; and Zhou, M. 2018. Neural latent extractive document summarization. In *Proceedings of EMNLP*.

Zhang, Y.; Zincir-Heywood, A. N.; and Milios, E. E. 2004. World wide web site summarization. *Web Intelligence and Agent Systems* 2(1):39–53.

Zhou, Q.; Yang, N.; Wei, F.; and Zhou, M. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of ACL*, 1095–1104.

Zhu, J.; Wang, Q.; Wang, Y.; Zhou, Y.; Zhang, J.; Wang, S.; and Zong, C. 2019. NCLS: Neural cross-lingual summarization. In *Proceedings of EMNLP*, 3045–3055.