Guiding Generation for Abstractive Text Summarization based on Key Information Guide Network

Chenliang Li and **Weiran Xu*** and **Si Li** and **Sheng Gao**
Beijing University of Posts and Telecommunications, Beijing
chenliangli,xuweiran,lisi,gaosheng@bupt.edu.cn

Abstract

Neural network models, 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. We propose a guiding generation model that combines the extractive method and the abstractive method. Firstly, we obtain keywords from the text by a extractive model. Then, we introduce a Key Information Guide Network (KIGN), which encodes the keywords to the key information representation, to guide the process of generation. In addition, we use a prediction-guide mechanism, which can obtain the long-term value for future decoding, to further guide the summary generation. We evaluate our model on the *CNN/Daily Mail* dataset. The experimental results show that our model leads to significant improvements.

1 Introduction

Text summarization aims to generate a brief summary from an input document while retaining the key information. There are two broad approaches to summarization: *extractive* and *abstractive*. *Extractive models* (Mihalcea and Tarau, 2004; Yasunaga et al., 2017) usually extract a few sentences or keywords from the source text, while *abstractive models* (Rush et al., 2015; Nallapati et al., 2016) generate new words and phrases that not in the source text to construct the summary.

Recently, inspired by the success of encoder-decoder model (Sutskever et al., 2014), abstractive summarization models (Nallapati et al., 2016; See et al., 2017) are able to generate the summaries with high ROUGE scores. While these models proved to be capable of capturing the regularities of the text summarization, they are hard to be controlled in the process of generation. Without external guidance, these models just get the source text as input and then output the summary, which certainly leads to a lack of key information.

Zhou et al. (2017) propose a selective gate network to retain 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. (2017) propose a pointer-generator model, which uses the pointer mechanism (Vinyals et al., 2015) 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. To address these problems, we combine the extractive model and the abstractive model and use the former one to obtain keywords as guidance for the latter one.

In this paper we propose a guiding generation model for abstractive text summarization. Firstly, we use a extractive method to obtain the keywords from the text. Then, we introduce a Key Information Guide Network (KIGN), which 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 (Bahdanau et al., 2014) and the pointer mechanism. In addition, we propose a novel prediction-guide mechanism based on He et al. (2017), which predicts the extent of key information covered in the final summary, to further guide the summary generation. Experiments show that our model achieves significant improvements.

2 Related work

Neural encoder-decoder models. Abstractive models(Rush et al., 2015; Chopra et al., 2016) have been widely used in text summarization. Nallapati et al. (2016) use a pointer network (Vinyals

* Corresponding Author: Weiran Xu

Proceedings of NAACL-HLT 2018, pages 55–60
New Orleans, Louisiana, June 1 - 6, 2018.

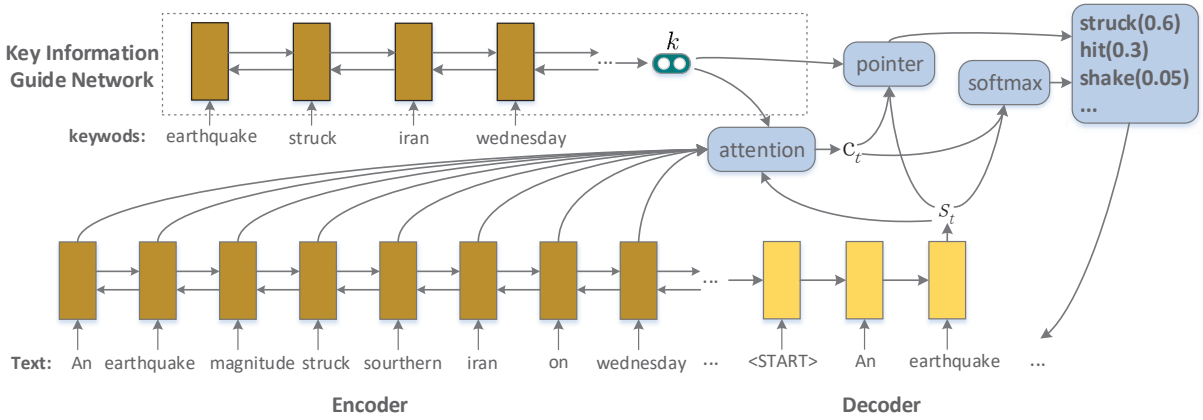

Figure 1: Our key information guide model. It consists of key information guide network, encoder and decoder. In the key information guide network, we encode the keywords to the key information representation k.

et al., 2015) to deal with the unknown word problem.

Keywords extraction. TextRank algorithm (Mihalcea and Tarau, 2004), which extracts keywords from the source text, is unsupervised.

Prediction-guide mechanism. Inspired by the success of AlphaGO, He et al. (2017) propose a prediction network to predict the long-term value of the final summary. Our prediction-guide mechanism is use to guarantee the more key information covered in the final summary.

3 Our Model

In this section, we describe (1) our baseline encoder-decoder model, (2) our key information guide network, and (3) our prediction-guide mechanism.

3.1 Encoder-decoder model based attention

Our baseline model is similar to that of Nallapati et al. (2016). The tokens of the input article $x = \{x_1, x_2, ..., x_N\}$ are fed into the encoder, which maps the text into a sequence of encoder hidden states $\{h_1, h_2, ..., h_n\}$. At each decoding time step t, the decoder reads the previous word embedding w_{t-1} and the previous context vector c_{t-1} as inputs to obtain the decoder hidden state s_t. The context vector c_t is calculated by using the attention mechanism:

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

$$\alpha_t^e = softmax(e_t) \quad (2)$$

$$c_t = \sum_{i=1}^{N} \alpha_{ti}^e h_i \quad (3)$$

where v, W_h, W_s are learnable parameters, h_i is the hidden state of the input token x_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:

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

where f represents a linear function.

3.2 Key information guide network

Most encoder-decoder models (Zhou et al., 2017; See et al., 2017) just get the source text as input and then output the summary, which is hard to be controlled in the process of generation and leads to a lack of key information in the summary. We propose a key information guide network to guide the process of generation from two aspects: the attention mechanism and the pointer mechanism.

In detail, we extract keywords from the text by using TextRank algorithm. As shown in Figure 1, the keywords are fed one-by-one into the key information guide network, and then we concatenate the last forward hidden state $\vec{h}_n$ and backward hidden state $\overleftarrow{h}_1$ as the key information representation k:

$$k = \begin{bmatrix} \overleftarrow{h}_1 \\ \vec{h}_n \end{bmatrix} \quad (5)$$

Attention mechanism: Traditional attention mechanism is hard to identify keywords, which just uses the decoder state as a query to get the attention distribution of the encoder hidden states. We use the key information representation k as

extra input to the attention mechanism, changing equation (1) to:

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

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

Our key information representation k makes the attention mechanism more focus on the keywords. That is seem like to introduce prior knowledge to the model.

Then, we apply the key information representation k and use the new context vector c_t to calculate a probability distribution over all words in the vocabulary, changing equation (4) to:

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

where v represents that y_t is from the target vocabulary.

Pointer mechanism: Due to the limitation of the vocabulary size, some keywords may not be in the target vocabulary, which will certainly lead to a lack of them in the final summary. Therefore we take the key information representation k, the context vector c_t and the decoder hidden state s_t as inputs to calculate a soft switch p_{sw}, which is used to choose between generating a word from the target vocabulary or copying a word from the input text:

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

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

Our pointer mechanism, which is equipped with the key information representation, has the ability to identify the keywords. We use the new attention distribution α_{ti}^e as the probability of the input token w_i and obtain the following probability distribution to predict the next word:

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

Note that if w is an out-of-vocabulary word, $P_v(y_t = w)$ is zero.

During training, we minimize a maximum-likelihood loss at each decoding time step, which is most widely used in sequence generation. We define y_t^* as the target word for the decoding time step t and the overall loss is:

$$L = -\frac{1}{T} \sum_{t=0}^{T} logP(y_t^*|y_1^*, ..., y_{t-1}^*, x) \quad (10)$$

3.3 Prediction-guide mechanism at test time

At test time, when predicting the next word, we consider not only the above probability (Equation 9), but also a long-term value predicted by the prediction-guide mechanism. The prediction-guide mechanism is based on He et al. (2017).

Our prediction-guide mechanism, which is a single-layer feed forward network with sigmoid activation function, predicts the extent of the key information covered in the final summary. At each decoding time step t, we take mean pooling over the decoder hidden states $\bar{s}_t = \frac{1}{t} \sum_{l=1}^{t} s_l$, the encode hidden states $\bar{h}_n = \frac{1}{n} \sum_{i=1}^{n} h_i$ and the key information representation k as inputs to calculate the long-term value.

We sample two partial summaries y_{p1} and y_{p2} for each x with random stop to get $\bar{s}_t$. Then, we finish the generation from y_p to obtain M average decoder hidden states $\bar{s}$ of the completed summaries $S(y_p)$ (using beam search), and compute the average score:

$$AvgCos(x, y_p) = \frac{1}{M} \sum_{\bar{s} \in S(y_p)} cos(\bar{s}, k) \quad (11)$$

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 $AvgCos(x, y_{p1}) > AvgCos(x, y_{p2})$. Therefore, the loss function of the prediction-guide network is as follows:

$$L_{pg} = \sum_{(x, y_{p1}, y_{p2})} e^{v(x, y_{p2}) - v(x, y_{p1})} \quad (12)$$

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

At test time, we first compute the normalized log probability of each candidate, and then linearly combine it with the value predicted by the prediction-guide network. In detail, given an abstractive model $P(y|x)$ (Equation 9), a prediction-guide network $v(x, y)$ and a hyperparameter $\alpha \in (0, 1)$, the score of partial sequence y for x is computed by:

$$\alpha \times logP(y|x) + (1 - \alpha) \times log\ v(x, y) \quad (13)$$

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

Model	ROUGE-1	ROUGE-2	ROUGE-L
Enc-dec+attn baseline (50k vocab)	31.33	11.81	28.83
Abstractive model (Nallapati et al., 2016)	35.46	13.30	32.65
Baseline+pointer	36.44	15.66	33.42
KIGN	37.76	16.56	34.49
Prediction-guide	37.24	16.27	34.14
KIGN+Prediction-guide	**38.95**	**17.12**	**35.68**

Table 1: ROUGE F1 scores for models on the CNN/Daily Mail test set. All our ROUGE scores have a 95% confidence interval of at most ±0.25 as reported by the official ROUGE script.

4 Experiments

4.1 Experiment setting

We use the *CNN/Daily Mail* dataset(Nallapati et al., 2016; Hermann et al., 2015) and use scripts supplied by Nallapati et al. (2016) to obtain the same version of the data, which has 28,7226 training pairs, 13,368 validation pairs and 11,490 test pairs. We use two 256-dimensional LSTMs for the bidirectional encoder and one 256-dimensional LSTM for the decoder. In our key information guide network, the approach of encoding keywords is same to the encoder. In addition, we use a vocabulary of 50k words for both source and target and do not pre-train the word embeddings - they are learned from scratch during training. During training and testing, we truncate the text to 400 tokens and limit the length of the summary to 100 tokens. We train using Adagrad (Duchi et al., 2011) with learning rate 0.15 and an initial accumulator value of 0.1. The batch size is set as 16. Following the previous work, our evaluation metric is F-score of ROUGE (Lin and Hovy, 2003).

In addition, for the prediction-guide mechanism, we set the single-layer feed forward network with 800 nodes. For the hyperparameter α, we test the performances of KIGN+Prediction-guide model using different α during decoding. As can be seen from the figure 2, the performance is stable for the α ranging from 0.8 to 0.95. When α is set as 0.9, we can obtain the highest F-score of ROUGE. Besides, we set the M as 8 and adapt mini-batch training with batch size to be 16. The network is trained with AdaDelta (Zeiler, 2012).

During training and at test time we truncate the input tokens to 400 and limit the length of the output summary to 100 tokens for training and 120 tokens at test time, which is similar to See et al. (2017). We trained our keywords network model less than $200,000$ training iterations. Then we trained the single-layer feed forward network based on the KIGN model. Finally, at test time, we combine the KIGN model and the prediction-guide mechanism to generate the summary.

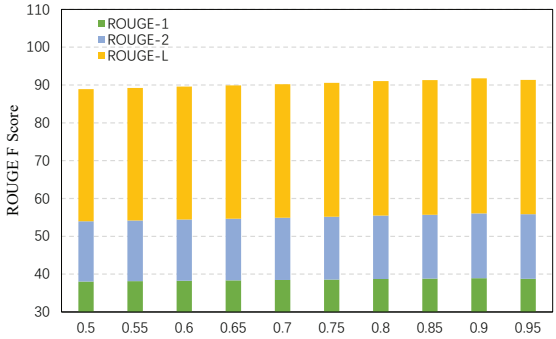

Figure 2: ROUGE-1, ROUGE-2 and ROUGE-L F1 scores of KIGN+Prediction-guide model w.r.t different hyperparameter α.

4.2 Results and discussions

We compare our model with the baseline model (enc-dec+attn), hierarchical networks (Nallapati et al., 2016) and the baseline model equipped with pointer-mechanism since we use the pointer mechanism in our model.

Table 1 shows that our key information guide network scores exceed the baseline model equipped with the pointer-mechanism by (+1.3 ROUGE-1, +0.9 ROUGE-2, +1.0 ROUGE-L). In addition, we just add the prediction-guide mechanism on the baseline model equipped with the pointer-mechanism to understand the contribution of each part. The scores of that exceed the baseline model equipped with the pointer-mechanism by (+0.8 ROUGE-1, +0.6 ROUGE-2, +0.7 ROUGE-L). Finally, combining the key information guide network and the prediction-guide mechanism, we achieve a better performance. Our best model scores exceed the baseline model with pointer-

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 can under 82 languages in 20 distinct scripts, and works with both printed and cursive writing input with or without a stylus.

Figure 3: 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)

mechanism by (+2.5 ROUGE-1, +1.5 ROUGE-2, +2.2 ROUGE-L). In this paper, we do not implement coverage mechanism in our model, which can greatly improve the score of ROUGE (See et al., 2017).

4.3 Case study

Figure 3 is an example to show the coverage of the key information between the text and the summary and the bold words are the key information of the text. We compare the output of two models and give the gold summary. It shows that the main idea of the text is about google handwriting input working on android handsets and some function introduction. The baseline model equipped with pointer-mechanism produces the summary, which just shows that google have cracked the problem of reading handwriting, while the summary generated by our model covers almost all the key information of the text.

5 Conclusion

In this work, we propose a guiding generation model for abstractive text summarization. We combine the extractive model and the abstractive model. Firstly, we use the extractive method to obtain keywords from the input text. Then, we introduce a key information guide network, which encodes the keywords to the key information representation, to guide the process of generation. In addition, we propose a prediction-guide mechanism to further guide the generation at test time. Experiments show that our model leads to significant improvements.

Acknowledge

We thank the anonymous 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).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98. Association for Computational Linguistics.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159.

Di He, Hanqing Lu, Yingce Xia, Tao Qin, Liwei Wang, and Tieyan Liu. 2017. Decoding with value networks for neural machine translation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 177–186. Curran Associates, Inc.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the*

North American Chapter of the Association for Computational Linguistics.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In *Proceedings of EMNLP 2004*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290. Association for Computational Linguistics.

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

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 452–462. Association for Computational Linguistics.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

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