



Main Point Generator: Summarizing with a Focus

Tong Lee Chung^{1,2(✉)}, Bin Xu^{1,2}, Yongbin Liu^{3(✉)}, and Chunping Ouyang³

¹ Department of Computer Science and Technology,
Tsinghua University, Beijing, China
tongleechung86@gmail.com, xubin@tsinghua.edu.cn

² Beijing National Research Center for Information Science and Technology
(BNRist), Beijing, China

³ College of Computing, University of South China, Hengyang, China
qingbinliu@163.com, ouyangcp@126.com

Abstract. Text summarization is attracting more and more attention while deep neural network has had many successful application in NLP. One problem of such models is its inability to focus on the essentials of documents, thus generating summaries that may not be important, especially during multi-sentence summarization. In this paper, we propose Main Pointer Generator (MPG) to address the problem, where at each decoder step the whole document is taken into consideration when calculating the probability of next generated token. We experiment with CNN/Daily news corpus and results show that summaries our MPG generated follow the main theme while outperforming the original pointer generator network by about 0.5 ROUGE point.

Keywords: Text summarization · Sequence-to-sequence · Pointer Coverage

1 Introduction

Text summarization is the task of producing a short piece of text from a long one while preserving main information [1]. *Extractive* and *abstractive* are the two general approaches to text summarization. Extractive methods focus on feature engineering for selecting words and phrases from the original text to assemble the summary. On the other hand, abstractive models may produce words that are not in the original text.

Recent advances in neural sequence to sequence [2] (encoder decoder) models, in which RNNs read and generate text with recurrent neural networks has made many advances in various NLP tasks such as machine translation, question answering as well as summarization [3, 4]. Multi-sentence summarization has attracted friction recently due to introduction of new techniques and datasets [5]. Standard sequence to sequence model is the baseline of neural network summarization, which opens new possibilities to abstractive summarization [6, 7].

State-of-the-art models have been presented by several researches where some problems have been tackled when using RNNs for abstractive summarization. Attention mechanism has improved quality by showing the decoder what part of the sentence to attend to [5]. High level attributes, such as length constraint, entity-centric and source-specific are introduced to control the generation of summaries [3]. Pointer-generator with coverage mechanism model [4] handles accuracy and out-of-vocabulary problem via pointing (copying words from the document) while retaining its ability to generate new words, and uses coverage to eliminate repetition. New learning objective have been introduced to optimize ROUGE with reinforcement learning [8].

Repetition is a problem these model face where summaries tend to repeat itself. Two ways of dealing with the repetition is presented, coverage/intra-attention of words [4, 8] and directly preventing generation of same n-grams [3, 8]. Coverage discourages generation of the same word by using a coverage vector that sums over attention received so far during decoding showing what words have been looked at already [4], similarly intra-attention can be seen as calculating a coverage over generated words [8]. While the other solution directly prevents the generation of n-grams during decoding [3]. Though it has solved the repetition problem and improved summary quality, it might be causing another problem: the summary not being able to focus on the main points of the document. When summarizing, important words such as names, places may appear multiple times, because they are the main focus of documents such as news. Anti-repetition strategies causes the decoder to penalize them after appearing even though they are key words, and instead generates summaries that are less important. Our aim is to produce summaries that are closely related to the main theme of the document.

While the task of summarization is to preserve main information of the document, current model focus on generating summaries that do not produce false facts or do not repeat itself. Most do not consider the fact that summaries should focus on the main theme of the document. In this paper, we present Main-pointer-Generator (MPG) model, which neutralizes the negative effect of coverage and helps the decoder focus on the key parts of the document. The model, derived from pointer-generator network with coverage [4], takes into consideration a document vector at each decoding step and can generates key words of the document multiple times. We experiment on *CNN/Daily Mail* dataset that is used by various researchers and outperform state-of-the-art model and is comparable to the result of lead-3 baseline (using first three sentences as summary). We also show that the summary our model produces focuses on one theme.

2 Related Work

Summarization has been around for many decades [9], this paper focuses on multi-sentence summarization where the task is to generate summaries with multiple sentences.

Neural Abstractive Summarization. The abstractive approach needs to understand the meaning of the document, and then briefly summarize it by a highly readable human language. For this target, the RNN/LSTM models and some of their deformations are adopted to complete the neural abstractive task [10]. Recently, some researchers have used the latest neural networks model to summarize, such as the sequence-to-sequence model and attention model.

Sequence-to-Sequence Models. Most of current state-of-the-art models are based on sequence-to-sequence models which have gained many successes in machine translation [2, 11]. Attention [12], pointer network [13], coverage [14] and controllable summarization [3, 15] are some techniques adapted to the task of summarization.

Most similar to our work is from Fan et al. [3]. Controllable summaries lets reader decide which entity they want their summary to focus on, giving them the option on which parts to focus on. An entity marker is used to instruct the decoder where to look, where the marker can be specified by human or automatically. Such approach requires entity recognition tool to find entities. Main difference between our work is that we rely on the whole document rather than entities to find the focus.

3 Main Pointer Generator Network Model

This section describes our Main Pointer Generator Network Model (MPG) in detail. The model is derived from Pointer-Generator Network from See et al. [4] and is depicted in Figure 1. We will go through vital parts of the whole model in this section.

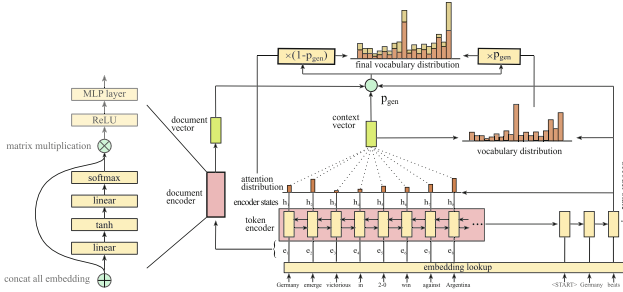


Fig. 1. Main pointer generator model. At each decoder timestep, the decoder state, context vector, and document vector is used to calculate a generation probability p_{gen} . Decoder state shows what words are already generated, the context vector shows where to attend to in the document, and the document vector helps focus on important content of the documents.

3.1 Encoding

Like most neural network model, tokens are embedded into fixed length vectors $E = (e_1, e_2, \dots, e_n)$. Embedding flows two direction, a token encoder and a

document encoder. The token encoder (a bi-directional LSTM network) produces two outputs, an output state for every input token $H = (h_1, h_2, \dots, h_n)$ and a final output state, which is used as the first input during the decoding process. The document encoder takes in the embeddings E and generates a fixed vector representing the whole document using self attending mechanism.

The document encoder contains a bi-directional LSTM (separate from the token encoder), that generates output states $H' = (h'_1, h'_2, \dots, h'_n)$ for each input embedding. A self attending mechanism is applied to the output states to compute a representation for the document. First attention distribution A^{doc} is calculated using $\tanh$ followed by a *softmax* operation, then multiplied with the state vector H' :

$$A^{doc} = \text{softmax}(W_{h2} \tanh(W_{h1} H')) \quad (1)$$

$$d' = H'^T A^{doc} \quad (2)$$

The multiplication gives a hidden state of the document d' with a fixed length eliminating the document size. Finally, ReLU and Multilayer Perceptron (MLP) is applied to generate a final output d with a specific size:

$$d^h = \text{ReLU}(h') \quad (3)$$

$$d = W_{output} d^h + b_{output} \quad (4)$$

W_{output} and b_{output} are both trainable parameters. By setting size of *output*, we can generate a document vector of a specific size. d is the final document vector and is calculated once for every document and shared during each decoding step.

3.2 Decoding

Now that the input document is encoded, three outputs are generated, a document vector d from document encoder, output states $H = (h_1, h_2, \dots, h_n)$ for each token and final output state from token encoder. We now show how summary is generated at each time step. A LSTM cell takes as input a linearity of previous context vector h_{t-1}^* (described later) and embedding of previous word e'_{t-1} (previous word from reference summary when training, and previous generated word during testing) and produces a current state s_t (at step 0, inputs are final output state from token encoder and pre-defined <START> token).

$$x_t = w_{l1} h_{t-1}^* + w_{l2} e'_{t-1} + b_l \quad (5)$$

$$s_t = \text{LSTM}(x_t) \quad (6)$$

where w_{l1} , w_{l2} and b_l are trainable parameters. Attention distribution is next, which can be calculated in two ways depending on whether coverage mechanism is used. When coverage is not used, attention distribution of a token a^t is calculated using a linear operation of its encoder output state h_i and decoder state s_t followed by nonlinearity $\tanh$ and *softmax*:

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + b_{attn}) \quad (7)$$

$$a^t = \text{softmax}(e^t) \quad (8)$$

W_h , W_s and b_{attn} are trainable parameters. When using coverage, a coverage vector c^t is maintained, which sums over all the previous attention distribution (at time step $t = 0$, c^0 is a zero vector because nothing is covered). Token attention distribution is calculated similar but includes c^t to the linear operation:

$$c^t = \sum_{t'=0}^{t-1} a^{t'} \quad (9)$$

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + W_c c_i^t + b_{attn}) \quad (10)$$

$$a^t = \text{softmax}(e^t) \quad (11)$$

newly included parameter W_c is trainable. The coverage vector reminds the current attention calculation which tokens were observed, thus avoiding attending to the same part of the documents. In the discussion section, we show why coverage may be the reason unimportant summary is generated. Attention is a probability over the source document telling the decoder where to look when generating words. A context vector h_t^* can now be calculated using the attention distributions and encoder states:

$$h_t^* = \sum_i a_i^t h_i \quad (12)$$

The context vector is a fixed-size representation the document based on the summary produced so far. It can be used together with the current decoder state to produce the vocabulary distribution by going through two linear and a softmax layer:

$$P_{vocab} = \text{softmax}_{vocab}(V'(V[s_t, h_t^*] + b) + b') \quad (13)$$

V , V' and b' are trainable parameters. Next token could be one with the highest probability, but this would cause misleading facts and inability to deal with out-of-vocabulary tokens. A pointer generator is then calculated to show which part of the documents it relates to and if the next token should be copied from the original document or generated by the distribution. Context vector h_t^* , current state s_t , decoder input x_t (linearity of previous word embedding and context vector), and document vector are all used to calculate a generation probability $p_{gen} \in [0, 1]$:

$$p_{gen} = \sigma(w_{h^*} h^* + w_s s_t + w_x x_t + w_d d + b_{ptr}) \quad (14)$$

where w_{h^*} , w_s , w_x , and w_d are trainable parameters, and σ is the sigmoid function. p_{gen} See et al. [4] only takes in w_{h^*} , w_s , and w_x , which means that when deciding to choose between generating and copying, only information from what have been generated is considered, the document as a whole is not considered. We adjust p_{gen} using the document vector d to focus on the main content even though appearing multiple times, thus getting the name Main Pointer Generator. d is shared during all decoding step. Using p_{gen} to re-calculate word probability

by considering between copying and generating, the word probability distribution $P(w)$ from which we predict words can be calculated:

$$P(w) = p_{gen}P_{vocab}(w) + (1 - p_{gen}) \sum_{i:w_i=w} a_i^t \quad (15)$$

3.3 Training

The loss function to train the model is the summation of negative log likelihoods of target word w^* at each time step:

$$loss_t = -\log P(w_t^*) \quad (16)$$

$$loss = \frac{1}{T} \sum_{t=0}^T loss_t \quad (17)$$

When using the coverage mechanism, a coverage loss is included to penalize repeatedly attending at the same words, making loss at each time step:

$$loss_t = -\log P(w_t^*) + \lambda \sum_i \min(a_i^t, c_i^t) \quad (18)$$

4 Experiments and Results

We present details of our experiment in this section, which includes the hyper-parameters and training of the model, the dataset used, evaluation method and results.

4.1 Model Parameters and Optimization

The model is implemented in Tensorflow and trained using a Titan X GPU. During training, we keep hyper-parameter same as See et al. [4] to show that our model produces higher scores not by fine tuning but by generating better summaries. Hidden state size is set to 256 and embedding size set to 128. Vocabulary size is 50 K. Pre-trained word embedding is not used during the training of the models. Parameters that are unique to our model are those related to the document vector. Hidden size for two non-linearity (tanh and softmax) for self attention is set to 350 and 30 respectively. Hidden size for ReLU layer is 3000. Finally, MLP layer has 256 hidden units to generate a document vector of size 256. The model is trained with AdaGrad with learning rate 0.15 and an initial accumulator value of 0.1. Gradient clipping with a maximum gradient norm of 2 is used, and do not use other forms of regularization. We follow the procedure that See et al. used by truncating the document to 400 tokens and limiting summary to 100 for training and 120 for testing. The model is trained in batches with size of 16. At test time, summaries are produced using beam search with beam size of 4.

4.2 Dataset and Evaluation

The CNN\Daily dataset is separated in training set, validation set and testing set with 287226, 13368 and 11490 pairs of training examples respectively. Summaries consist of 3.75 sentences on average. Our model is evaluated with the standard ROUGE metric [16] and we report F1 score for ROUGE-1, ROUGE-2, and ROUGE-L. We compare with the original pointer-generator + coverage model [4] and controllable model [3], the only two models we can find that experimented on the original text dataset. These two are the current state-of-the-art models for multi-sentence summarization. We also compare with lead 3 baseline (first three sentences used as summary), a hard to beat baseline. We also perform qualitative analysis by looking at examples generated by our model and the original pointer generator + coverage model.

4.3 Results

This section presents quantitative and qualitative experiment results.

Table 1. Results for experiment. Our model MPG + coverage has the highest ROUGE score.

	ROUGE 1	ROUGE 2	ROUGE L
Lead-3 baseline (See et al. [4])	40.34	17.70	36.57
Pointer-generator(See et al. [4])	36.44	15.66	33.42
Pointer-generator + coverage (See et al. [4])	39.53	17.28	36.38
Controllable abstractive (Fan et al. [3])	39.75	17.29	36.54
MPG (Ours)	37.84	16.50	34.88
MPG + coverage, shared layer (ours)	39.69	17.36	36.42
MPG + coverage (Ours)	40.12	17.74	36.82

Table 1 show the results for our experiments. We perform experiment using three models of our own, MPG without coverage mechanism, MPG with a shared LSTM layer and our full MPG model. We see that coverage does produce a boost giving a extra +2.28 on ROUGE 1, +1.24 on ROUGE 2, and +1.94 on ROUGE L. We also see that sharing a LSTM layer between two encoders does not perform better. But on the other hand, there are less parameters to train faster. We can see also that by using the document vector, ROUGE score has improved compared to the original pointer generator. When using coverage, our model performed better with +0.59 on ROUGE 1, +0.46 on ROUGE 2, and 0.44 on ROUGE L. So we can say that using a document vector can increase ROUGE performance by 0.5 ROUGE points. We also compare with controllable abstractive summary with automatic control and find that our model performs better. We can also see that our results are the only model that is closest to the lead-3 baseline.

5 Conclusion

This paper, we investigate a problem of focusing during summarization. We analyze a state-of-the-art pointer generator + coverage model and conclude reason for such the problem. We propose a model that makes the generate summaries based of the main theme of the document. Experiment is conducted on CNN\Daily dataset and we achieve state-of-the-art results increasing ROUGE point by 0.5. This work serves as a stepping stone of neural network summarization as our technique can be integrated into other models.

Acknowledgments. This work is supported by China National High-Tech Project (863) under grant (No. 2015AA015401). Beijing Key Lab of Networked Multimedia also supports our research work. The work is supported by State Key Program of National Natural Science of China (No. 61533018), National Natural Science Foundation of China (No. 61402220), and the Philosophy and Social Science Foundation of Hunan Province (No. 16YBA323).

References

1. Nenkova, A., McKeown, K.: Automatic summarization. *Found. Trends Inf. Retrieval* **5**(23), 103–233 (2011)
2. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS 2014* (2014)
3. Fan, A., Grangier, D., Auli, M.: Controllable abstractive summarization. *CoRR*, abs/1711.05217 (2017)
4. See, A., Liu, P.J., Manning, C.D.: Get to the point: summarization with pointer-generator networks. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics* (2017)
5. Nallapati, R., Zhou, B., dos Santos, C.N., Gülçehre, Ç., Xiang, B.: Abstractive text summarization using sequence-to-sequence rnns and beyond. In: *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, 11–12 Aug 2016*
6. Chopra, S., Auli, M., Rush, A.M.: Abstractive sentence summarization with attentive recurrent neural networks. In: *The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016, San Diego California, USA, 12–17 June 2016*
7. Rush, A.M., Chopra, S., Weston, J.: A neural attention model for abstractive sentence summarization. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, 17–21 Sept 2015*
8. Paulus, R., Xiong, C., Socher, R.: A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304 (2017)
9. Saggion, H., Poibeau, T.: Automatic text summarization: past, present and future. In: Poibeau, T., Saggion, H., Piskorski, J., Yangarber, R. (eds.) *Multi-source, Multilingual Information Extraction and Summarization. Theory and Applications of Natural Language Processing*, pp. 3–21. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-28569-1_1

10. Shen, S., Zhao, Y., Liu, Z., Sun, M.: Neural headline generation with sentence-wise optimization. *arXiv Computation and Language* (2016)
11. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: *International Conference on Learning Representations (ICLR)* (2015)
12. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) *Advances in Neural Information Processing Systems*, vol. 30, pp. 6000–6010. Curran Associates Inc. (2017)
13. Vinyals, O., Fortunato, M., Jaitly, N.: Pointer networks. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) *Advances in Neural Information Processing Systems*, vol. 28, pp. 2692–2700. Curran Associates Inc. (2015)
14. Mi, H., Sankaran, B., Wang, Z., Ittycheriah, A.: Coverage embedding models for neural machine translation. In: *EMNLP* (2016)
15. Kikuchi, Y., Neubig, G., Sasano, R., Takamura, H., Okumura, M.: Controlling output length in neural encoder-decoders. *CoRR*, abs/1609.09552 (2016)
16. Lin, C.-Y.: Rouge: a package for automatic evaluation of summaries. In: *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*