VAE-PGN based Abstractive Model in Multi-stage Architecture for Text Summarization

Hyungtak Choi[1], Lohith Ravuru[1], Tomasz Dryjański[2], Seonghan Ryu[1], Donghyun Lee[1], Hojung Lee[1] and Inchul Hwang[1]
[1] Samsung Research, Samsung Electronics Co. Ltd., Seoul, Korea
[2] Samsung R&D Institute, Warsaw, Poland
{ht777.choi, loki.ravuru, t.dryjanski, seonghan.ryu, dh.semko.lee, hojung76.lee, inc.hwang}@samsung.com

Abstract

This paper describes our submission to the TL;DR challenge. Neural abstractive summarization models have been successful in generating fluent and consistent summaries with advancements like the copy (Pointer-generator) and coverage mechanisms. However, these models suffer from their extractive nature as they learn to copy words from the source text. In this paper, we propose a novel abstractive model based on Variational Autoencoder (VAE) to address this issue. We also propose a Unified Summarization Framework for the generation of summaries. Our model eliminates non-critical information at a sentence-level with an extractive summarization module and generates the summary word by word using an abstractive summarization module. To implement our framework, we combine submodules with state-of-the-art techniques including Pointer-Generator Network (PGN) and BERT while also using our new VAE-PGN abstractive model. We evaluate our model on the benchmark Reddit corpus as part of the TL;DR challenge and show that our model outperforms the baseline in ROUGE score while generating diverse summaries.

1 Introduction

Text summarization is the task of producing an accurate summary by preserving essential information from a long text document. This is a challenging and significant task, as it can be applied to many real-world applications such as summarizing news articles, social media, web pages, blogs or long text documents. Many approaches have been proposed to solve the text summarization problem (See et al., 2017; Liu, 2019b; Gehrmann et al., 2018). Extractive summarization generates a summary by selecting important phrases or sentences from the source text. This approach mainly uses ranking the importance of phrases or sentences to select only important information (Liu, 2019b). Whereas the abstractive approach generates entirely new phrases or sentences that capture the meaning of the source text. In this paper, we focus on a few challenges that need to be addressed in generating an abstractive summarization of a given text.

The first challenge comes from the difficulty in preserving the contents of a large source text in the generated summary. The state-of-the-art models for abstractive summarization use a sequence-to-sequence attention model with a copy mechanism (Gu et al., 2016; See et al., 2017), to select the relevant content of the source text. But these models suffer from their extractive nature of generating summaries due to the copy mechanism (Boutkan et al., 2019; Chawla et al., 2019). We introduce a VAE-based PGN model to overcome this extractive nature. Another challenge is to eliminate non-critical information from the source text. Gehrmann et al. (2018) employ a word-level content selection model to focus on only critical information, but handling critical information at word-level is difficult in long sentences because of the repetition of many common words. Our approach handles the critical information by using a multi-stage model with a sentence-level selection (extractive) and abstractive summarization modules.

First, we use a fine-tuned BERT-based extractive model named BERTSUM (Liu, 2019b) to eliminate less important sentences by scoring each sentence in the source text. Second, an abstractive summary is generated on the basis of the extracted sentences by the extractive model. This combination of VAE and PGN mechanisms brings diversity to abstractive summaries. This sequential multi-stage processing improves performance compared to using single-stage abstraction models. We found that the proposed model performs well, achieving the best result compared to our

Proceedings of The 12th International Conference on Natural Language Generation, pages 510–515,
Tokyo, Japan, 28 Oct - 1 Nov, 2019.

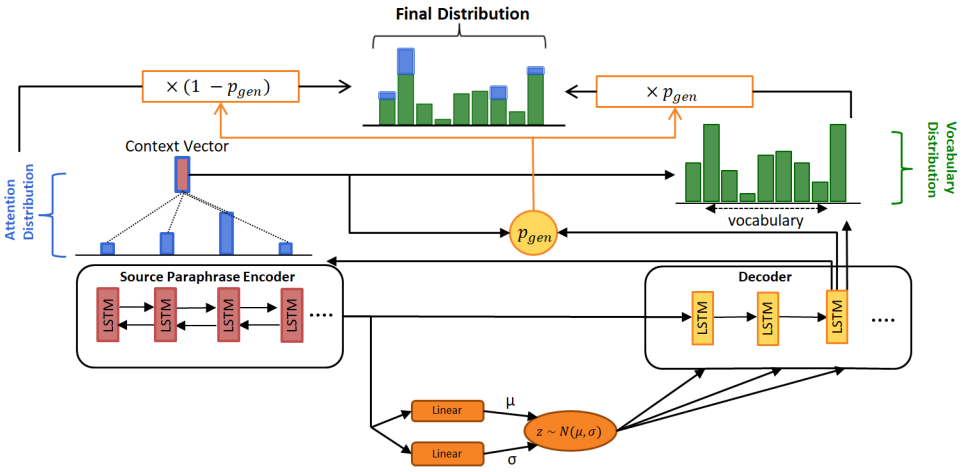

Figure 1: Overview of VAE-PGN, combining a pointer-generator network based encoding for copying the content related words and a VAE-based encoding for generating the variations.

baselines on this task.

The contributions of our work are as follows:

- We develop VAE-based PGN to address the diversity issue and generate abstractive summaries.
- We propose a new architecture that excludes less important information at the first stage.

2 Related Work

2.1 Extractive Summarization

Zhang et al. (2018) use a latent variable extractive model where sentences are viewed as latent variables and sentences with activated variables are used to infer gold summaries. Dong et al. (2018) utilize a policy gradient reinforcement learning algorithm to select a sequence of sentences that maximize ROUGE scores. Zhou et al. (2018) use a novel end-to-end neural network framework for extractive document summarization by jointly learning to score and select sentences. Motivated by BERT (Devlin et al., 2018), which has achieved state-of-the-art performance on multiple NLP tasks, Liu (2019b) use a fine-tuned BERT model to score the sentences. This model is used by our work for extractive summarization module.

2.2 Abstractive Summarization

The work on attention-based encoder-decoder models (Rush et al., 2015; Chopra et al., 2016) created a surge of research interest in text generation approaches. Recent abstractive summarization models adopt the attention mechanism (Bahdanau et al., 2015), and a pointer-network to copy infrequent words and entities to the target sentence (See et al., 2017; Gülçehre et al., 2016; Nallapati et al., 2016; Gu et al., 2016). Gehrmann et al. (2018) use content selector that can be used for a bottom-up attention that restricts the ability of abstractive summarizers to copy words from the source text.

2.3 Diversity

In this work, the diversity of summarization refers to generating words that are different from the source text. VAEs have become more and more popular (Hu et al., 2017; Shen et al., 2017) in generating diverse sentences through learning a high-level latent variable representation of the context. Bowman et al. (2015) points out that generating sentences from a continuous space like VAE is of higher grammatical quality compared to other techniques such as Beam Search.

3 Proposed Architecture

In this section, we explain the proposed architecture to resolve the problems discussed above.

3.1 VAE-PGN Model

As illustrated in Figure 1, the proposed model is based on two main components, the pointer-generator network, and the VAE mechanism. We incorporate the vanilla VAE to learn a representation of source text that captures the complex semantic structures underlying the text. The latent representation from the VAE component is combined with the pointer-generator component to generate the summary at the decoder.

An LSTM encoder encodes source text. From the final state of the encoder, we sample a latent variable over a Gaussian distribution. An LSTM decoder is used to generate the summary. Each decoding step uses the latent variable from the encoder and an attention mechanism to generate a probability distribution over the vocabulary. A pointer-generator network is used in tandem to aid in copying words from the source text.

Now we define the notations used in the rest of the paper. We represent the training data as $\{s_i, t_i\}_{i=1}^{N}$. It contains N content-summary pairs. $s = (w_1^s, w_2^s, ..., w_p^s)$ represents source text of length p, and $t = (w_1^t, w_2^t, ..., w_q^t)$ represents the target summary of length q. The source and target embeddings of a word w_i, are represented as $e_{w_i}^s$ and $e_{w_i}^t$ respectively.

The embeddings of words from the source text are encoded by a bidirectional LSTM Encoder.

$$h^s = ENC_e^s(e_{w_1}^s, e_{w_2}^s, e_{w_3}^s, ..., e_{w_p}^s) \quad (1)$$

where ENC_e^s is the bidirectional LSTM encoder and h^s is the final encoder state of the source text. Since we use a bi-directional encoder, the hidden states of forward and backward encoder are concatenated, and passed through a feed-forward network to get h^s. The final state of encoder is passed through two feed-forward layers to get μ and σ respectively.

$$\mu = Linear_1(h^s) \text{ and } \sigma = Linear_2(h^s) \quad (2)$$

Although $Linear_1$,$Linear_2$ layers are both feed-forward layers, they are labelled separately here to reflect that the weights are updated differently during backpropagation. The latent variable which encodes the representation of the source sentence is modelled as a one-dimensional tensor whose dimensions is a hyperparameter. We set this hyperparameter is initialized to dimensions of the encoder and decoder. A random variable from the $N(0, I)$ normal distribution is sampled and is transformed to the required $N(\mu, \sigma)$ distribution.

$$z^{'} \sim N(0, I) \longrightarrow z \sim N(\mu, \sigma) \quad (3)$$

The LSTM decoder is initialized with the final state of the encoder h^s. The input to the decoder at every time step is $z \oplus e_{w_t}$, which is a concatenation of the latent variable z and embeddings of the previous word (e_{w_t}). During training, e_{w_t} is the embedding of the previous word of target paraphrase and while testing, it is embedding of the word generated by the decoder at step t-1.

The loss function is the sum of the cross-entropy loss calculated between the target and the generated summary, the Coverage loss, and the Kullback-Leibler Divergence (KL-D) loss calculated between the N(0, I) distribution and the generated $N(\mu, \sigma)$ distribution. The loss function can be formalized as follows:

$$L(\theta, \phi; x^s, x^t) = E_{q_\phi(z|x^s)}[logp_\theta(x^t|z, x^s)] + \lambda \sum_i min(a_i^t, c_i^t) - KL(q_\phi(z|x^s)||p(z)) \quad (4)$$

3.2 Multi-Stage Architecture

Gehrmann et al. (2018) studies the effect of eliminating duplicate or insignificant words by performing a word-level content selection. Motivated by the content selection, we use a sentence level selection before generating an abstractive summary as described by the Algorithm 1.

Algorithm 1 Multi-Stage Architecture

1: Train two language models, one using BERTSUM for extractive summarization and the other using PGN or VAE-PGN for abstractive summarization.
2: Get scores for sentences from the source text using BERTSUM
3: Reorder the sentences using Algorithm 2
4: Generate abstractive summary using the Abstractive model (PGN or VAE-PGN)

We obtain a score for each sentence through a high-performance extractive summarization model based on fine-tuned BERT called BERTSUM (Liu, 2019b). The scored sentences are then reordered using the Algorithm 2. Also, the number of input sentences for the abstractive summary was selected dynamically according to the parameter min_words in Algorithm 2. This hyperparameter is tuned based on the average target summary length of a given dataset. As a result, abstractive summarization can be performed with the filtered input in which the unimportant sentences are removed.

4 Experiments

4.1 Datasets and Experimental Setup

TL;DR Reddit corpus (Völske et al., 2017): This is the dataset for the TL;DR challenge. They pro-

Algorithm 2 Order Preserving Selection

$A = list(< sentence, id, score >)$
procedure REORDER(A)
  $sortedA = sortByScore(A)$
  $maxcount = getmax(sortedA)$
  $trimA = sortedA[0 : maxcount]$
  $reorderedA = sortById(trimA)$
  return $reorderdA$
end procedure
procedure GETMAX(A)
  for $< sentence, id, score >$ in A **do**
    $words$ += no. of words in $sentence$
    $maxcount$ += 1
    if $words > min\_words$ **then**
      break
    end if
  end for
  return $maxcount$
end procedure

vide a corpus consisting of approximately 3 million content-summary pairs mined from Reddit. The competition organizers split the dataset into training, validation and test datasets.

CNN/Daily Mail corpus (Hermann et al., 2015; Nallapati et al., 2016): We use the non-anonymized version (See et al., 2017) of the dataset which contains pairs of online news articles and their summaries. The dataset contains approximately 287,000 training pairs, 13,368 validation pairs, and 11,490 testing pairs.

The VAE implementation of paper Bowman et al. (2015), Gavrilov (2017) and the PGN implementation of paper See et al. (2017), Kumar (2019) are used as references for abstractive module. The BERTSUM implementation of paper Liu (2019b), Liu (2019a) is used as a reference for the extractive module.

Extractive Summarization and Abstractive Summarization modules are finetuned on each datasets for obtaining respetive results. Apart from the minimum words generated, we borrow all hyperparameters from baseline code implementations. The min_words parameter is tuned by setting it to the average target summary length and doing to a grid-search based fine-tuning around the initialized value.

4.2 Metrics and Baselines

ROUGE (Lin, 2004) metric is used as an automatic evaluation metric. Rouge-N (R-n) scores refer to the n-gram overlap between the generated and the reference summary. Rouge-L (R-L) score refers to the Longest Common Subsequence based overlap score. All scores are calculated using the virtual machine provided by competition (2018) on the TIRA platform that uses the tagucci (2019) project to calculate the scores. We compared models using a validation set, choose the best scored models and then evaluated them on the official test set due to resource and time constraints.

PGN is the pointer-generator model with copy and coverage mechanism. **VAE-PGN** is the VAE-based pointer-generator model with copy and coverage mechanism as shown in Figure 1. **Unified PGN** and **Unified VAE-PGN** refer to the Unified Architecture with PGN and VAE-PGN models used for respective abstractive summarization modules.

4.3 Results

Table 1: Results on CNN-DM test dataset

Model	R-1	R-2	R-L
PGN	38.86	16.70	35.38
VAE-PGN	39.06	16.93	**35.61**
Unified PGN	39.00	16.65	29.08
Unified VAE-PGN	**39.32**	**17.07**	29.43

The results on the test set of CNN/Daily Mail dataset are presented in Table 1. The results show that VAE-PGN model performs better than the PGN model by achieving higher ROUGE scores. Although the unified model increases the R-1 and R-2 scores on both the abstractive models, there is a reduction in R-L score. We expect this decline to be caused by using different tokenization techniques used in the abstractive model (uses Stanford NLP tokenizer) and the extractive model (uses BERT tokenizer).

Table 2: Results on Reddit data validation set

Model	R-1	R-2	R-L
PGN	18	3	13
VAE-PGN	18	3	12
Unified PGN	19	4	**15**
Unified VAE-PGN	**20**	4	14

Table 3: Results on Reddit data test set

Model	R-1	R-2	R-L
Unified PGN	19	4	15
Unified VAE-PGN	19	4	15

The results on the validation set and the test set are presented in Table 2 and Table 3 respectively. The results show that the unified architecture model outperforms the abstractive model on both VAE and VAE-PGN based models and also the Unified VAE-PGN model gets a better R-1 score than Unified PGN on the validation set. But Unified PGN and Unified VAE-PGN model perform almost similarly on the test set.

5 Conclusions

In this paper, we propose a Unified VAE-PGN model and an effective multi-stage architecture for abstractive summarization. Our model eliminates non-critical information at sentence-level and also generates diverse summaries using a continuous space representation of the information. We evaluate our models on the benchmark Reddit datasets as part of the TL;DR challenge and show that our proposed models outperform the baseline models. We plan to include content selection, eliminating word-level non-critical information in the multi-stage architecture in future work.

Acknowledgments

The authors would like to thank Katarzyna Beksa and Katarzyna Podlaska for their helpful review and suggestions.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Freek Boutkan, Jorn Ranzijn, David Rau, and Eelco van der Wel. 2019. Point-less: More abstractive summarization with pointer-generator networks. *CoRR*, abs/1905.01975.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *CoRR*, abs/1511.06349.

Kushal Chawla, Kundan Krishna, and Balaji Vasan Srinivasan. 2019. Improving generation quality of pointer networks via guided attention. *CoRR*, abs/1901.11492.

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, San Diego, California. Association for Computational Linguistics.

TL;DR competition. 2018. Tl;dr the abstractive summarization challenge.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. 2018. Banditsum: Extractive summarization as a contextual bandit. *CoRR*, abs/1809.09672.

Daniil Gavrilov. 2017. pytorch_rvae.

Sebastian Gehrmann, Yuntian Deng, and Alexander M. Rush. 2018. Bottom-up abstractive summarization. *CoRR*, abs/1808.10792.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *CoRR*, abs/1603.06393.

Çaglar Gülçehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *CoRR*, abs/1603.08148.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *CoRR*, abs/1506.03340.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Controllable text generation. *CoRR*, abs/1703.00955.

Atul Kumar. 2019. pytorch_rvae.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yang Liu. 2019a. Bertsum.

Yang Liu. 2019b. Fine-tune BERT for extractive summarization. *CoRR*, abs/1903.10318.

Ramesh Nallapati, Bing Xiang, and Bowen Zhou. 2016. Sequence-to-sequence rnns for text summarization. *CoRR*, abs/1602.06‘023.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *CoRR*, abs/1509.00685.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. *CoRR*, abs/1704.04368.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. 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 6830–6841. Curran Associates, Inc.

tagucci. 2019. pythonrouge.

Michael Völske, Martin Potthast, Shahbaz Syed, and Benno Stein. 2017. TL;DR: Mining Reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 59–63, Copenhagen, Denmark. Association for Computational Linguistics.

Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. 2018. Neural latent extractive document summarization. *CoRR*, abs/1808.07187.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. *CoRR*, abs/1807.02305.