A Normalized Encoder-Decoder Model for Abstractive Summarization Using Focal Loss

Yunsheng Shi, Jun Meng, Jian Wang(✉), Hongfei Lin, and Yumeng Li

Dalian University of Technology, Dalian 116023, Liaoning, China
wangjian@dlut.edu.cn

Abstract. Abstractive summarization based on seq2seq model is a popular research topic today. And pre-trained word embedding is a common unsupervised method to improve deep learning model's performance in NLP. However, during applying this method directly to the seq2seq model, we find it does not achieve the same good result as other fields because of an over training problem. In this paper, we propose a normalized encoder-decoder structure to address it, which can prevent the semantic structure of pre-trained word embedding from being destroyed during training. Moreover, we use a novel focal loss function to help our model focus on those examples with low score for getting better performance. We conduct the experiments on NLPCC2018 share task 3: single document summary. Result showed that these two mechanisms are extremely useful, helping our model achieve state-of-the-art ROUGE scores and get the first place in this task from the current rankings.

Keywords: Summarization · Seq2Seq · Pre-trained word embedding
Normalized encoder-decoder structure · Focal loss

1 Introduction

Summarization is the task to compress a piece of text to a shorter version that contains the main ideal of the original. There are two main approaches to summarization: extractive and abstractive. Extractive method is taking some sentences directly from the source text. While Abstractive method is generating novel words and sentences not featured in source text. Abstractive is more difficult than extractive because it transforms the source text to summary in human-like style, which require its incorporation of real-world knowledge. Recently, due to the success of seq2seq model with attention mechanism [1, 2], abstractive approaches get greatly development and most of studies are based on this model [3–5].

However, although these systems are promising, they are based on large-scale corpus, which may not perform well in small dataset. NLPCC2018 share task 3 is a single-document-summary task with small training dataset, which only contains 50,000 articles with summary. To get the better performance on this small corpus, we try to use the pre-trained word embedding like [4] but do not get the corresponding improvement

M. Zhang et al. (Eds.): NLPCC 2018, LNAI 11109, pp. 383–392, 2018.
https://doi.org/10.1007/978-3-319-99501-4_34

because of that the semantic structure of pre-trained word embedding is very easy to be destroyed during training, causing the model to diverge (see Fig. 2 from Sect. 2).

After many attempts, we present a novel normalized encoder-decoder structure to solve this problem, in which we add a normalization layer in both encoder and decoder to prevent the semantic structure of pre-trained word embedding from being damaged by over training, which has an incredible effect in our experiments.

What's more, inspired by the reference [6], we also identify the same sample imbalance problem in text summarization that there is a small part of examples from the whole dataset the model difficultly fits well and gets low score on it. we propose a novel focal loss function to force our model pay more attention on those hard examples during training for getting better performance. This loss function is from above paper and has never been used in text summarization.

We apply our models in NLPCC2018 share task 3, and the experimental results show that these two mechanisms are extremely useful and help us achieve state-of-the-art ROUGE scores, getting the first place in this task finally.

2 Our Models

In this section, we describe (1) baseline sequence-to-sequence model (2) copy and coverage mechanisms (3) our normalized encoder-decoder structure and focal loss.

2.1 Sequence-to-Sequence Attentional Model

Our baseline model is similar to that of [7]. After being segmented, Chinese sentence was transformed to tokens w_s_i, which will be fed into the encoder (a single-layer bidreaction LSTM), producing a sequence of encoder hidden state h_s_i, abstractly computed as:

$$h\_s_i = f(h\_s_{i-1}, w\_s_i) \tag{1}$$

Where f computes the current hidden state given the previous hidden state h_s_{i-1} and source tokens w_s_i and can be either an Elman RNN unit, a GRU, or a LSTM unit. In this paper, we use LSTM unit as encoder and decoder.

During decoding, the structure is the same as encoder in addition to an input of last hidden state h_t_{t-1}, abstractly computed as:

$$h\_t_{\mathrm{t}} = f(h\_t_{\mathrm{t}-1}, [w\_t_{t-1}; h\_t_{t-1}]) \tag{2}$$

Them, an attentional hidden state is produced as follows:

$$\tilde{h}_t = \tanh(W_{\mathrm{c}}[c_t; h\_t_t]) \tag{3}$$

$$e_i^t = v^T \tanh(W_a[h\_t_t; h\_s_i] + b_a) \tag{4}$$

$$a^t = \text{softmax}(e^t) \tag{5}$$

$$c_t = \sum_i a_i^t h\_s_i \tag{6}$$

At first, we calculate the attention distribution a_i as the concat attention in [7] (4) (5). Next, the attention distribution is used to produce a weighted sum of the encoder hidden states, known as the context vector c_t (6). At last, an attentional hidden will be calculate by (3), where the W_c is learnable parameter.

Finally, we produce the vocabulary distribution P_{vocab} by $\tilde{h}$:

$$p_{vocab} = \text{softmax}(W_{vocab}\tilde{h} + b_{vocab}) \tag{7}$$

During training, the loss for time step t is the negative log likelihood of the target word's vocabulary probability $P_{vocba}(w\_t_t)$ for that time step:

$$loss_t = -\log p_{vocab}(w\_t_t) \tag{8}$$

2.2 Copy and Coverage Mechanisms

Pointer-Generator Network

To deal with the out-of-vocabulary (OOV) words problem, we take the pointer-generator network, which was proposed from [5] as copy mechanism (see Fig. 1). A copy probability $P_{copy} \in [0,1]$ will control the model whether generate the target word from P_{vocab} (7) or from a^t.

$$p_{\text{copy}} = \sigma(W_{\tilde{h}}^T \tilde{h} + b_{copy}) \tag{9}$$

Where $\tilde{h}$ is an attentional hidden state calculated by Eq. (3).

The final word distribution is presented as follow:

$$P(w) = p_{copy}P_{vocab} + (1 - p_{copy}) \sum_{w_i=w} a_i^t \tag{10}$$

Note that if the p_{copy} is zero, the model will copy the word from the ith word of source sequence text whose a_i^t is the Maximum value in a^t. Otherwise, if p_{copy} is one, The model will generate target word from vocabulary distribution.

The loss function is the same as Eq. (8) by replace p_{vocab} with $P(w)$.

Coverage Mechanism

We also apply coverage mechanism [5] to prevent the model generate the same word repeatedly. In each decoder step t, we will calculate a coverage vector

$$co^t = \sum_{k=1}^{t-1} a^k \tag{11}$$

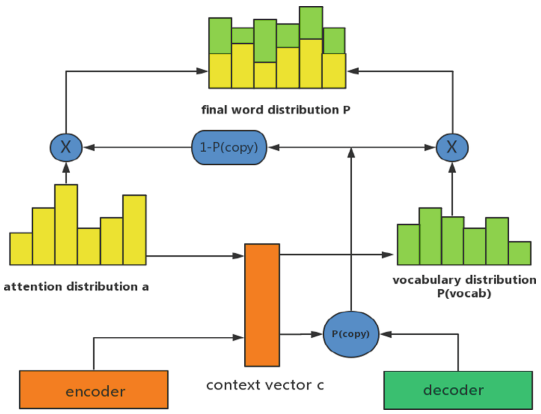

Fig. 1. Copy mechanism in the model. In the decoder step t, the final $P(w)$ is calculated by the sum of attention distribute a^t and vocabulary distribution P_{vocab}

The co^t will be used to prevent each word from being generated more than once by the penalty loss as follow:

$$loss^t_{\text{cov}} = \sum_{k=1}^{L} \min(co^t_k, a^t_k) \tag{12}$$

Where the L is the length of source text.

Finally, the total loss function for training the model is the weighted sum of negative log-likelihood $P(w)$ (10) and coverage loss (12), and the λ will be set to 1 as hyperparameter:

$$Loss = \frac{1}{T}\sum_{t=1}^{T}(-\log P(w_t) + \lambda loss^t_{\text{cov}}) \tag{13}$$

2.3 Normalized Encoder-Decoder Structure and Focal Loss

Normalized Encoder-Decoder Structure

Conventional seq2seq abstractive models are built for training on the large-scale corpus such as CNN/Daily Mail dataset (with 287,113 training examples) [8] and Gigaword corpus (with 5 million examples) [9], which may cause their hardly performing well on small datasets. We try to use pre-training word embedding such as word2vec [10] to deal with this problem.

However, during our experiments, we find that directly loading the pre-trained word embedding to model cannot make a good result, even make the model eventually diverge because of over training problem (see Fig. 2).

The semantic structure of pre-trained word embedding is very easy to be destroyed during training the whole seq2seq model even if fixed it before training.

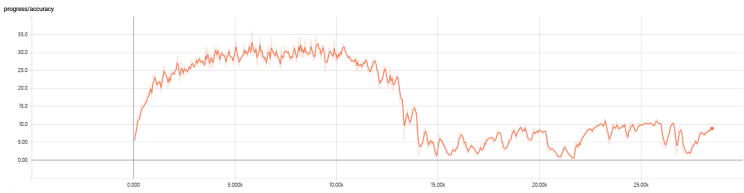

Fig. 2. The accuracy curve of baseline model with pre-trained word embedding.

After several trials, we find that using normalization layers can protect the semantic structure of word embedding from being destroyed during training, which is a new feature and never have been proposed in NLP.

Normalization layer was proposed for accelerating model's training. In this paper, we find it have ability of keeping the semantic structure of word embedding. we propose a normalized encoder-decoder structure (see Fig. 3). We add a normalization layer [11] between RNN and Embedding layer, apply this structure in both encoder and decoder. The result (see Fig. 5 from the Sect. 4) shows that our model is effectively prevent the model from diverging and maximizing the performance of pre-trained word embedding.

$$x_{norml} = \gamma \frac{x - \bar{x}}{\sqrt{\sigma_x^2 + \varepsilon}} + \beta \tag{14}$$

In Eq. (14), $\bar{x}$ is the mean value of x, σ_x^2 is the variance, γ and β are learnable parameters.

Focal Loss

Inspired by the approach in reference [6], there may have the same sample imbalance problem in text summarization. The training difficulty of each example in the training data is not the same. Moreover, the training difficulty of each word in a sentence (example) is also not the same, which can be represented by the $P(w)$ from Eq. (10). The $P(w)$ means that a word is an easy training example where its $P(w)$ closed to 1 because the model can predict it with confidence of one hundred percent, while it will be a difficultly training word if its $P(w)$ is small to zero.

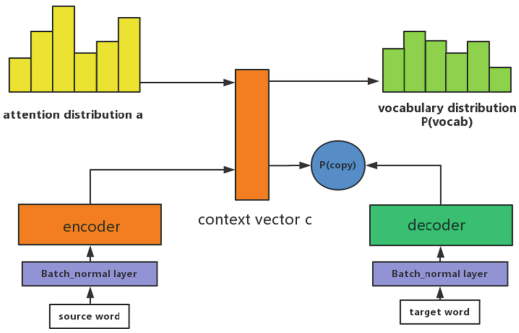

Fig. 3. Normalization layer will normalize the word vector before being put in RNN network, which will be applied in both encoder and decoder.

Consequently, we need to assign a soft weight to each word's loss for the model can pay more attention on those whose $P(w)$ is small for get better performance on it. The focal loss for it will be presented like follow:

$$loss_{focal\_loss}(w_t) = \alpha(1 - P(w_t))^{\gamma}(-\log(P(w_t)) \tag{15}$$

Where the $\alpha(1 - P(w_t))^{\gamma}$ is the weight of $loss_w$. When the $P(w_t)$ is small, the $loss_{focal\_loss}$ of this word will be enlarged by its weight, which will lead the model optimizing on this word's loss more. While if the $P(w_t)$ is large to one, the $loss_{focal\_loss}$ will be close to zero, and it does not provide any help for the final optimization.

The α and the γ are hyperparameters, we will set them for 0.25 and 1 for respectfully in this experiment.

The final loss will calculate by $\text{loss}_{\text{focal\_loss}}$ and coverage loss (12) like this:

$$Loss = \frac{1}{T}\sum_{t=1}^{T}(loss_{focal\_loss}(w_t) + \lambda loss_{\text{cov}}^{t}) \tag{16}$$

3 Dataset and Experiments

3.1 TTNews Corpus

We training our model on the TTNews corpus provided by NLPCC2018 share task 3. We use all the news to pre-trained the word embedding. We take 5,000 news with summary as training set. We take 1,239 examples from the NLPCC2017 test set (because there are some examples in training set redundantly) as validation set. Finally, we will predict the summary for the NLPCC2018 test set (with 2,000 examples).

3.2 Experiments

We do with two main experiments. Firstly, we train the following five models (1) baseline model, (2) baseline model + word2vec, (3) baseline model + fixed word2vec, (4) normalized encoder-decoder model, (5) normalized encoder-decoder model + word2vec, comparing their accuracy to prove the validity of our model.

Secondly, we will take the best model from the first result, changing its NLL (negative log likelihood) into Focal Loss to do contrast experiment.

For all experiments, our basic seq2seq models have 600-dimensional hidden states bid-LSTM encoder and 1200-dimensional hidden states LSTM decoder. And each encoder and decoder only have 1 layer. We use a vocabulary of 60k words for both source and target and set embedding size as 400, and all the models are using copy and coverage mechanisms.

What's more, We use Adam as our optimizing algorithm by default hyperparameters (learning rate $\alpha = 0.001$, two momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ respectively, and $\varepsilon = 10^{-8}$).

During training and at test time we do not truncate the article or the summary. We train on a single Titan XP GPU with a batch size of 2. At test time our summaries are produced using beam search with beam size 5.

We trained all models for about 100,000 iterations, because we will get the best model in the first 4 epochs for preventing over fitting. Each epoch take about an hour.

We tried different truncated text to train the model, but we found it will reduce the model performance. We try different α and γ of FL and found that the model with FL will outperform than with NLL in the same learning rate when $\alpha = 0.25$ and $\gamma = 1$.

4 Validation and Result

4.1 Accuracy Evaluation

For quickly evaluating training model's performance on validation set, we count a accuracy, which is obtained by dividing the number of words appearing both in prediction summary and in target summary by total number of target summary words. The equation is presented as follow, in which the n is the number of examples.

$$\text{acc} = \frac{1}{n}\sum_{i=1}^{n}\frac{number(w_{predict} \cap w_{target})}{number(w_{target})} \tag{17}$$

We draw the training accuracy curve of baseline model+word2vec and of normalized encoder-decoder model+word2vec with the iteration numbers as X-axis and the accuracy as Y-axis (see Figs. 4 and 5).

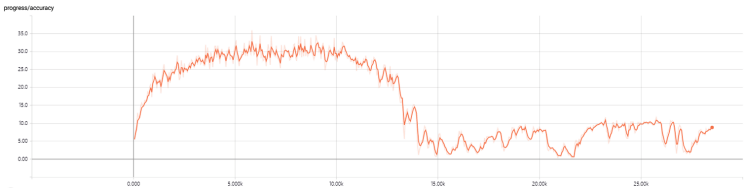

Fig. 4. The training accuracy curve of baseline model with pre-trained word embedding.

Fig. 5. The training accuracy curve of normalized encoder-decoder model with pre-trained word embedding.

Figures 4 and 5 show that the accuracy curve of baseline model with pre-trained word embedding will be shock to zero suddenly after 15,000 iterations. While for normalized encoder-decoder model, it is improving stably with the increase of iteration. Consequently, it is no doubt that our normalized encoder-decoder model is truly useful for keeping the semantic structure of pre-trained word embedding from being destroyed during training.

Moreover, to prove the superiority of our model, we train the five models and gets their accuracy on validation set after each epoch. The result is showed as Fig. 6.

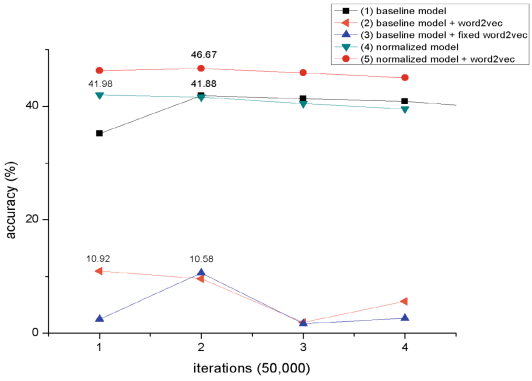

Fig. 6. The accuracy curves of the following five models on validation set: (1) baseline model, (2) baseline model+word2vec, (3) baseline model+fixed word2vec, (4) normalized encoder-decoder model, (5) normalized encoder-decoder model+word2vec.

There are three conclusions we can get from above Fig. 6. Firstly, directly loading the pre-trained embedding to baseline model will lead a terrible result that model will diverge during training even if fixed it (see curve (2) and curve (3)). Secondly, simply using normalized encoder-decoder model without pre-trained word embedding will not bring a significant improvement to result (see curve (1) and curve (4)). Finally, as we see, the normalized encoder-decoder model with pre-trained word embedding get a very high accuracy on validation set, which is at least 5% higher than others.

Consequently, there is a reason to believe that our model can maximum the performance of pre-trained word embedding.

We also evaluate normalized encoder-decoder model + word2vec with NLL (negative log likelihood) and with FL (focal loss) on validation (see Fig. 7). The model with FL is about 2% accuracy higher than that with NLL, in which we can learn that FL can really improve the model's performance compared with NLL.

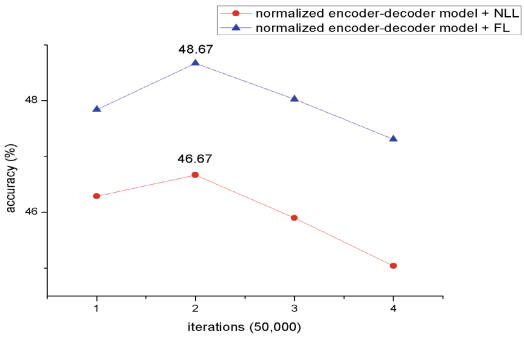

Fig. 7. The accuracy curves for normalized encoder-decoder model with NLL and with FL

4.2 ROUGE

We evaluate our models with the standard ROUGE metric [12]. We report ROUGE-F2, ROUGE-F4 and ROUGE-SU* for our validation set (with 1,239 examples) by the official ROUGE script (version 1.5.5) as NLPCC2018 share task 3 require.

We also evaluate our models through online evaluation. NLPCC2017 share task 3 testset: https://www.biendata.com/competition/nlptask03/ and NLPCC2018 share task 3 testset: https://biendata.com/competition/nlpcc2018/.

Showed as the follow Table 1, the performance of each model evaluated by ROUGE is the same as evaluated by Accuracy.

Table 1. ROUGE-F from validation set and average ROUGE from NLPCC2017 and NLPCC2018 online evaluation

Models	ROUGE-F2	ROUGE-F4	ROUGE-SU*	Testset 2017	Testset 2018
(1) baseline model	0.48764	0.18675	0.64069	0.30524	/
(2) baseline model+word2vec	0.14633	0.01136	0.14545	0.08666	/
(3) baseline model+fixed wor2vec	0.14425	0.01129	0.14396	0.08592	/
(4) normalization encoder-decoder model + NLL	0.49177	0.18714	0.64607	0.30544	/
(5) normalization encoder-decoder model + word2vec + NLL	0.50220	0.19615	0.65490	0.31650	/
(6) normalization encoder-decoder model + word2vec + FL	**0.51779**	**0.20895**	**0.66309**	**0.32450**	**0.31292**

Our model with focal loss achieve much better scores than others in each ROUGE points (Despite there is some discrepancy between the data we evaluate and the online assessment because of the lack of the specific value of rouge script's parameter using in online evaluation).

What's more, our best model (NLPCC2018_DLUT_815) is also the best model in NLPCC2018 share task 3 competition, getting the **31.292** average ROUGE scores, which is **1.3** points higher than the second place.

5 Conclusion

In this work, we presented a normalized encoder-decoder model to maximize the effect of pre-trained word embedding. What's more, a focal loss we take to help model fit those difficultly training examples better. The experiments proved these two mechanisms had obviously improved the performance of seq2seq model applying in text summarization.

Finally, we applied our model on NLPCC2018 share task 3: single document summary, achieving state-of-the-art ROUGE scores and getting the first place from the current rankings.

Acknowledgments. This research is supported by the National Key Research Development Program of China (No. 2016YFB1001103).

References

1. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Neural Information Processing Systems (2014)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: International Conference on Learning Representation (2014)
3. 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, pp. 379–389 (2015)
4. Nallapati, R., Zhou, B., dos Santos, C., Gulcehre, C., Xiang, B.: Abstractive text summarization using sequence-to-sequence RNNs and beyond. In: Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, pp. 280–290 (2016)
5. 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, vol. 1, pp. 1073–1083 (2017)
6. Lin, T.-Y., Goyal, P., Girshick, R., He, K.: Focal loss for dense object detection (2018). arXiv preprint: arXiv:1708.02002
7. Luong, M.-T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: Empirical Methods on Natural Language Processing (2015)
8. Hermann, K.M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., Blunsom, P.: Teaching machines to read and comprehend. In: Advances in Neural Information Processing Systems, pp. 1693–1701 (2015)
9. Graff, D., Kong, J., Chen, K., Maeda, K.: English gigaword. Linguistic Data Consortium, Philadelphia (2003)
10. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. CoRR, abs/1310.4546 (2013)
11. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Learning Representation (2015)
12. Lin, C.-Y.: Rouge: a package for automatic evaluation of summaries. In: Text Summarization Branches Out: ACL Workshop (2004b)