Attempting to Improve Pointer-Generator Networks for Abstractive Summmarization

Aviv Rosenberg [1] **Alon Frydberg** [1]

Abstract

Pointer-Generator Networks have performed well in the task of abstractive text summarization for long documents. In this paper we experiment with several approaches in attempt to improve the results of the basic pointer-generator architecture. Our attempts are evaluated on the newly published NEWSROOM dataset using the ROUGE metric. One attempt - a stacked encoder with a much lower learning rate - performed exceptionally well, considerably improving the state-of-the-art single layer encoder model published with the dataset by 5.01 ROUGE-1 , 6.3 ROUGE-2, 4.08 ROUGE-L.

1. Introduction

Summarization is the task of condensing a piece of text to a shorter version that contains the main information from the original. There are two broad approaches to summarization: extractive and abstractive. Extractive methods assemble summaries exclusively from passages (usually whole sentences) taken directly from the source text, while abstractive methods may generate novel words and phrases not featured in the source text – as a human-written abstract usually does.

The current state-of-the-art model for summarization is pointer-generator (PG) networks (See et al., 2017). This is a hybrid model that facilitates copying words from the source text via pointing (Vinyals et al., 2015), which improves accuracy and handling out-of-vocabulary (OOV) words, while retaining the ability to generate new words. The network also uses a coverage mechanism (Tu et al., 2016) to track and control coverage of the source document.

In this paper we attempt to experiment with methods and features that, to the best of our knowledge, were not yet used in the architecture and training of PG networks:

- Predict the length of the summary, and crop the summary generated by the PG network to the predicted length.
- Use a multi-layer LSTM encoder.
- Use GAN - Add a discriminator network that tries to discriminate a reference summary from a generated summary. This network's prediction is added to the loss of the generator as a term of regularization, in order to produce summaries that resemble reference summaries "visually".
- Use pre-trained word embeddings.
- As a side effect, we also studied the influence of coverage and transfer learning (Yosinski et al., 2014).

We apply our features to the recently-introduced NEWSROOM dataset (Grusky et al., 2018), which contains 1.3 million news articles and human-written summaries. While most of them failed to improve existing results we show that the two-layer stacked encoder outperforms the single-layer one by at least 4 points in all 3 ROUGE (Lin, 2004) metrics.

2. Model

Our features are built upon the Pointer-Generator architecture, which we describe next.

2.1. Pointer-Generator Architecture

In the PG architecture, The tokens of the article w_i are fed one-by-one into the encoder (a single-layer bidirectional LSTM), producing a sequence of encoder hidden states h_i. On each step t, the decoder, a single-layer unidirectional LSTM, receives the word embedding of the previous word (while training, it is the previous word of the reference summary; and at test time it is the previous word emitted by the decoder), and has decoder state s_t. The attention distribution a^t is calculated as in Bahdanau et al. (2014):

$$e_i^t = v^T \tanh\left(W_h h_i + W_s s_t + b_{atn}\right) a^t = softmax(e^t)$$

Next, the attention distribution is used to produce a weighted sum of the encoder hidden states, known as the context

[1]Department of Computer Science, Tel Aviv University, Tel Aviv, Israel. Correspondence to: Aviv Rosenberg <avivros007@gmail.com>, Alon Frydberg <alonfrydberg@mail.tau.ac.il>.

vector $h_t^\star$:

$$h_t^\star = \sum_i a_i^t h_i$$

The context vector is concatenated with the decoder state s_t and fed through two linear layers to produce the vocabulary distribution P_{vocab}. In addition, the generation probability $p_{gen} \in [0,1]$ for timestep t is calculated from the context vector $h_t^\star$, the decoder state s_t and the decoder input x_t:

$$p_{gen} = \sigma\left(w_{h^\star}^T h_t^\star + w_s^T s_t + w_x^T x_t + b_{ptr}\right)$$

Next, p_{gen} is used as a soft switch to choose between generating a word from the vocabulary by sampling from P_{vocab}, or copying a word from the input sequence by sampling from the attention distribution a^t. For each document let the extended vocabulary denote the union of the vocabulary, and all words appearing in the source document. The following probability distribution over the extended vocabulary is obtained:

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

During training, the loss for timestep t is the negative log likelihood of the target word $w_t^\star$ for that timestep:

$$loss_t = -\log P(w_t^\star)$$

and the overall loss for the whole sequence is:

$$loss = \frac{1}{T} \sum_{t=1}^{T} loss_t$$

In the coverage model, a coverage vector c^t is maintained, which is the sum of attention distributions over all previous decoder timesteps. The coverage vector is used as extra input to the attention mechanism, changing its calculation:

$$e_i^t = v^T \tanh\left(W_h h_i + W_s s_t + w_c^T c_i^t + b_{attn}\right)$$

The coverage loss to penalize repeated attention to the same locations is defined by:

$$covloss_t = \sum_i \min\{a_i^t, c_i^t\}$$

Finally, the coverage loss, reweighted by some hyper parameter λ, is added to the primary loss function to yield a new composite loss function:

$$loss_t = -\log P(w_t^\star) + \lambda covloss_t$$

Next, we show the features and methods used in our experiments.

2.2. Our Models

The code for our models is available online. [1]

2.2.1. Predicting Summary Length

After examining many summaries generated by the PG network, it was clear that almost all generated summaries have the same length (the maximal length), in spite of the network's ability to stop the generation at any timestep. This made the summaries appear unnaturally and unnecessarily long, which damages the network's precision. We attempted to fix that by predicting the summary's length from the article, and cropping the generated summary at the predicted length.

In order to achieve that goal, we train a separate network that predicts the length of an article's summary, we then use that length to crop the summary generated by the original PG network.

The length predicting network has a single-layer bidirectional LSTM encoder, similar to the one of the original PG model, producing a sequence of encoder hidden states h_i. The final hidden state h_n is fed through two linear layers to produce a single number ℓ predicting the length of the summary. The loss is simply the L_2 distance between the predicted length ℓ and the reference length $\ell^\star$.

2.2.2. Stacked Encoder

RNNs have shown to benefit from the depth of multi-layering (Pascanu et al., 2013). This method has not yet been extensively studied in the context of text generation. We hypothesized that this model would benefit greatly from the vast amount of data found in the NEWSROOM dataset.

We added a hyper-parameter ℓ_{layers} that determines the number of layers the encoder should have. We then created a bidirectional LSTM RNN of ℓ_{layers} with the same cell settings as in the single layer encoder. We ran three training sessions, with two, three, and four layers.

2.2.3. GAN Regularization

We used the GAN approach (Goodfellow et al., 2014) as a mean of regularization.

We are attempting to increase ROUGE scores. The ROUGE metric mostly relies on shared n-grams, and does not examine how similar two summaries are in substance. Therefore, even though the primary goal is for the network to generate accurate summaries of a given article, it must also generate summaries that are similar to the reference summaries, and share the same words.

[1] https://github.com/avivros007/SummarizationNEWSROOM

Thus, generating summaries as indistinguishable as possible from reference summaries might be beneficial.

In addition to the PG network G we added a discriminator network D. This network's goal is to discriminate weather its input is a reference summary or a generated summary (it outputs a probability that the input is a reference summary). We train the networks alternatively in the following manner.

For a given article art the GAN loss of the generator network will be

$$L_G = -D(G(art))$$

and this loss will be added to the general loss, multiplied by some hyper parameter λ_{GAN}.

For a given pair (art, ref) of article and reference summary the loss of the discriminator network will be

$$L_D = D(G(art)) - D(ref)$$

The discriminator network we used includes a single-layer bidirectional LSTM encoder, similar to the one of the original PG model, producing a sequence of encoder hidden states h_i. Then the final hidden state h_n is fed through one linear layer (with sigmoid activation) to produce a single number p which is the predicted probability for the input to be a reference summary.

2.2.4. Pre-trained Word Embeddings

High dimension word embeddings carry a lot more information than lower dimension ones. In an attempt to improve results, we wanted to increase the word embeddings dimension from 128 to 300, but the training time with such high dimension was not feasible for us.
Pre-trained word embeddings came to mind, as they offer the benefit of already training for months on a very large corpus, and also has the benefit of speeding up our training since the network doesn't need to train word embeddings at all.

Although pre-trained word embeddings proved to be much more beneficial for classification tasks (Kim, 2014), we wanted to see how well it performs our generation task.

2.2.5. Transfer Learning

We tried to take the PG network that was trained on the CNN / Daily Mail dataset (Hermann et al., 2015), and use it as the initial values for training the network on the Newsroom dataset. Our intuition was that these pre-trained network weights contain valuable information that will be useful as a starting point for the summarization task in the Newsroom dataset.

2.2.6. Coverage

In the original paper of pointer-generator networks (See et al., 2017) , after training has finished they added coverage and kept training for a short period of time. And so after evaluating without coverage, we also applied coverage and evaluated again.

3. Dataset and Evaluation

We use the Newsroom dataset which contains 1.3 million online news articles (658 words on average) paired with human-written summaries (26 words on average). It is a much larger dataset than previously published datasets, such as CNN/Daily Mail.

We evaluate our models with the ROUGE metric, reporting the F_1 scores for ROUGE-1, ROUGE-2 and ROUGE-L ,which respectively measure the word-overlap, bigram-overlap, and longest common subsequence between the reference summary and the summary to be evaluated. We obtain our ROUGE scores using the pyrouge package.

In section 5 we present our results. The ROUGE scores of a model will be denoted by (x, y, z) where x is its ROUGE-1 score, y is its ROUGE-2 score and z is its ROUGE-L score.

4. Training

Hyper-parameter tuning – For the first attempt, we simply took the PG network code published in See et al.(2017) and trained it on the Newsroom dataset. It did not converge at all on multiple attempts. After fully analyzing the model, and noticing that the training loss was oscillating, we concluded the learning rate, that was initially set to 1.5, was too high. We tried various learning rates, and in the end found that a much smaller 0.03 learning rate gave much better results on the Newsroom dataset, surpassing Gruski et al.'s (2018) scores in the 3 ROUGE metrics.

For all experiments, our model has 256-dimensional hidden states and 128-dimensional word embeddings, except for the model with pre-trained word embeddings which has 300-dimensional word embeddings. We use a vocabulary of 50k words, but for the model with pre-trained word embeddings we use a vocabulary of 100k words. We train using Adagrad (Duchi et al., 2011) with learning rate 0.03 and an initial accumulator value of 0.1. We use gradient clipping with a maximum gradient norm of 2, but do not use any form of regularization. For the coefficient of the GAN loss we found 0.05 to give the best results, and size of the trainable vocabulary in the model with pre-trained word embeddings 8k gave the best results.

Pre-trained word embeddings – In the attempt we chose to use GloVe (Pennington et al., 2014) pre-trained word embeddings, and specifically the *Wikipedia 2014 + Gigaword 5 GloVe*, available online[2]. The reason we used that particular one is that it is consisted of Wikipedia pages and news articles which resemble the NEWSROOM dataset. These are 300 dimension pre-trained word embeddings of 400k words. We experimented with the parameters. The first experiment was to use the 278,000 overlapping words with the words that appear in the NEWSROOM dataset. The second experiment was to additionally train the top 40k words by frequency that do not appear in the pre-trained embeddings dictionary. In the the third experiment we used a smaller dictionary of 100k from which 92k were pre-trained.

Curriculum learning – We used curriculum learning (Bengio et al., 2009) for the training process: At each stage we increase the maximal number of steps of both the encoder and the decoder. In appendix B you can find the exact curriculum we used. During test time we truncate the article to 400 words and limit the length of the summary to 35 words. This turned out to give the best results.

We train on a single TITAN Xp GPU with a batch size of 16. At test time our summaries are produced using beam search with beam size 4. Training a model took about $3-4$ days, while evaluating it took another $3-4$ days.

5. Results and Discussion

Our results are given in appendix A.

In addition to our own models, we also report the lead-3 baseline (which uses the first 3 sentences)(Grusky et al., 2018), and compare to the PG network (See et al., 2017) trained on the full training set of the NEWSROOM dataset and evaluated on the released test set of NEWSROOM, as reported by Gruski et al. (2018).

Gruski et al.'s (2018) PG network obtained ROUGE scores of $(26.02, 13.25, 22.43)$. The same model, trained with our hyper-parameter tuning, already greatly outperformed these results, achieving ROUGE scores of $(29.66, 18.23, 25.47)$.

The 2-layer encoder with the tuned hyper-parameters performed the best, achieving ROUGE scores of $(31.03, 19.55, 26.51)$. It outperformed Gruski et al.'s (2018) results by 5.01 R-1 , 6.30 R-2, 4.08 R-L scores.
The biggest increase relatively is of R-2, scoring 19.55 as opposed to 13.25 - 147% increase.

The extractive lead-3 model is extremely strong, achieving a score of $(30.49, 21.27, 28.42)$. In both See et al.(2017) and Gruski et al. (2018), the models don't surpass lead-3 in any ROUGE score, and are far behind in all three. Our stacked model surpasses lead-3 in R-1, and is very close in R-2 and R-L.

[2]https://nlp.stanford.edu/projects/glove/

Since the 2-layer encoder performed so well, the idea of a multi-layer encoder proved to be effective. However, the 3-layer and 4-layer encoders performed very poorly. Our theory is that there was a gross overfitting due to the high expressiveness of the hypotheses class. We believe this may be fixed with dropout or some other form of regularization.

Another method that performed well was transfer learning, which achieved slightly higher results than Gruski et al.'s (2018) PG baseline. However it still underperformed compared to the PG network with randomly initialized weights. This might be due to the fact that the network parameters converged to a local minima on the CNN/Daily Mail dataset.

The other methods (that underperformed):

- Using GAN - obtained decent results, close to the baseline of Gruski et al. (2018). The generator wasn't able to fool the discriminator as much as we hoped, which leads us to believe this model doesn't have much potential.
- Pre-trained word embeddings - did not work, and wasn't very surprising. Training of the word embeddings seems to be a crucial step for text generation to work. This is also indicated by the fact that the majority of the trainable parameters are in the embedding matrix.
  It did however shorten the training time as we expected.
- Cropping the length - did not work, but we believe that it might have the most potential. Our belief is based on the fact that even though the F_1-score dropped, the precision increased as we predicted.
  To further improve this method, one might use a smarter cropping system. Instead of a blunt crop, we could use the predicted length to integrate a soft switch that increases the probability that the decoder stops at a length close to the predicted one.
- Coverage - As opposed to the original paper, coverage did not work for some reason. We don't understand why, but it significantly lowered the results.

We are convinced that a combination of our methods, along with further fine tuning of hyper-parameters will yield even better results.

6. Conclusion

In this work we presented a few new features and changes to the pointer-generator architecture and training process. We applied our changes to a new and challenging dataset, and outperformed state-of-the-art models. While most of our approaches failed, we still believe some of them may be able to achieve better results with some further work.

References

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 41–48, New York, NY, USA, 2009. ACM.

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

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, pp. 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

Grusky, M., Naaman, M., and Artzi, Y. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. *CoRR*, abs/1804.11283, 2018.

Hermann, K. M., Kočiský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, pp. 1693–1701, Cambridge, MA, USA, 2015. MIT Press.

Kim, Y. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

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

Pascanu, R., Gülçehre, Ç., Cho, K., and Bengio, Y. How to construct deep recurrent neural networks. *CoRR*, abs/1312.6026, 2013.

Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pp. 1532–1543, 2014.

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

Tu, Z., Lu, Z., Liu, Y., Liu, X., and Li, H. Coverage-based neural machine translation. *CoRR*, abs/1601.04811, 2016.

Vinyals, O., Fortunato, M., and Jaitly, N. Pointer networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, pp. 2692–2700, Cambridge, MA, USA, 2015. MIT Press.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, pp. 3320–3328, Cambridge, MA, USA, 2014. MIT Press.

Appendices

A. Results

Full results in table 1. All our models (which are in the second part of the table) are the basic PG-networks with hyper-parameter tuning and added methods, and so we only write the added methods.
Except for the first entry that was trained on the CNN/Daily Mail dataset, all other entries were trained on the NEWSROOM dataset.
We have four entries in the table that use a pre-trained PG model by See et al. (2017), trained on the CNN/Daily Mail dataset, which is available online[1]. In the table we reference it by *PG-prev*.

Table 1. Full results of our experiments. All evaluations are on the NEWSROOM released test set. The first part of the table includes previous results or evaluations by a previously trained model. The second part are evaluations of models trained by us. The rows in **bold** are our best PG-based best performing model, and Gruski et al.'s (2018) PG baseline, which was also their best performing model.

MODEL	ROUGE-1	ROUGE-2	ROUGE-L
PG-prev (SEE ET AL., 2017) EVALUATED BY US ON NEWSROOM	23.02	10.44	18.62
LEAD-3 BASELINE (GRUSKY ET AL., 2018)	30.49	21.27	28.42
PG NETWORK BASELINE (GRUSKY ET AL., 2018)	**26.02**	**13.25**	**22.43**
INITIALIZED WITH *PG-prev* (SEE ET AL., 2017) WEIGHTS	26.52	14.73	22.57
INITIALIZED RANDOMLY	29.66	18.23	25.47
PG-prev (SEE ET AL., 2017) WITH LENGTH PREDICTION	21.98	10.08	18.30
INITIALIZED WITH *PG-prev* WEIGHTS, WITH LENGTH PREDICTION	12.52	3.60	10.55
WITH GAN LOSS	25.15	13.14	21.68
2-LAYER ENCODER	**31.03**	**19.55**	**26.51**
3-LAYER ENCODER	18.50	11.38	16.13
4-LAYER ENCODER	17.05	10.12	14.61
INITIALIZED RANDOMLY WITH COVERAGE	17.82	8.33	14.77
2-LAYER ENCODER WITH COVERAGE	19.11	8.85	15.79
WITH PRE-TRAINED WORD EMBEDDINGS	15.25	5.50	12.46

[1] https://github.com/abisee/pointer-generator

B. Curriculum Training

We trained the model with curriculum learning as shown in table 2.
If coverage was used, it was added to the model at the end of training, and training was resumed for approximately three additional hours.

Table 2. Curriculum learning stages

STAGE	MAX NUMBER OF ENCODER STEPS	MAX NUMBER OF DECODER STEPS	APPROXIMATE TIME (HOURS)
1	10	5	1
2	30	10	2
3	50	15	4
4	100	20	6
5	200	25	10
6	300	30	14
7	400	35	14