Text Summarization with Pointer-Generator Networks

Dareen Hussein, Israa Fahmy
PMDL Course – AUC Spring 2021

ABSTRACT

Abstractive text summarization now has a feasible new approach thanks to neural sequence-to-sequence models. When dealing with long text, however, they face challenges of inefficiency and accuracy: their capacity is insufficient to manage very long input, they cannot reliably replicate factual facts, and they tend to repeat themselves. That's why we suggest an extractive and abstractive hybrid model in this paper. We also look at how the neural network determines which input terms to pay attention to, and we pinpoint the functions of the different neurons in a simplified attention mechanism. Surprisingly, our simplistic attention mechanism outperforms the more complex attention mechanism on a set of papers that have been held out. We apply our model to the Amazon Fine-food reviews, outperforming the current abstractive state-of-the-art by at least 2 ROUGE points.

Keywords

LSTM, Seq2Seq, ROUGE, BLEU, Pointer-Generator, Coverage Mechanism.

1. INTRODUCTION

With the rise of the internet, we now have information readily available to us. We are bombarded with it literally from many sources — news, social media, office emails to name a few. If only someone could summarize the most important information for us! Deep Learning is getting there. In the big data era, there has been an explosion in the amount of text data from a variety of sources. This volume of text is an inestimable source of information and knowledge which needs to be effectively summarized to be useful. This increasing availability of documents has demanded exhaustive research in the NLP area for automatic text summarization. Automatic text summarization is the task of producing a concise and fluent summary without any human help while preserving the meaning of the original text document. Text summarization is very challenging, because when we as humans summarize a piece of text, we usually read it entirely to develop our understanding, and then write a summary highlighting its main points. Since computers lack human knowledge and language capability, it makes automatic text summarization a very difficult and non-trivial task.

2. MOTIVATION

Since manual text summarization is a time expensive and generally laborious task, the automatization of the task is gaining increasing popularity and therefore constitutes a strong motivation for academic research. There are important applications for text summarization in various NLP related tasks such as text classification, question answering, legal texts summarization, news summarization, and headline generation. Moreover, Customer reviews can often be long and descriptive. Analyzing these reviews manually, is really time-consuming. This is where we can apply Natural Language Processing to generate a summary for long reviews.

https://github.com/IsraaFahmy/PMDL/tree/main/Text%20summarizer%20for%20amazon%20reviews

3. LITREATURE RIVEW

3.1 Abstractive Summarization

Rush et al. (2015) were the first to apply modern neural nets to abstractive text summarization, achieving state-of-the-art results on two sentence-level summarization datasets, DUC-2004 and Gigaword. Their attention-based approach has been supplemented with recurrent decoders (Chopra et al., 2016), Abstract Meaning Representations (Takase et al., 2016), hierarchical networks (Nallapati et al., 2016), variational autoencoders (Miao and Blunsom, 2016), and direct optimization of the performance metric (Ranzato et al., 2016), all of which have improved performance on the datasets.

Large-scale datasets for summarising longer texts, on the other hand, are uncommon. Nallapati et al. (2016), is the one who adapted the DeepMind question-answering dataset (Hermann et al., 2015) for summarization and presented the first abstractive baselines. The same authors then published a neural extractive method (Nallapati et al., 2017), in which hierarchical RNNs are used to pick sentences, and found that it outperformed their abstractive result in terms of the ROUGE metric. These are the only two reported findings on the entire dataset that we are aware of.

3.2 Pointer-Generator

The pointer network (Vinyals et al., 2015) is a sequence-to-sequence model that employs Bahdanau et al. (2015) soft attention distribution to generate an output sequence made up of elements from the input sequence. The pointer network has been used to develop hybrid NMT (Gulcehre et al., 2016), language modelling (Merity et al., 2016), and summarization approaches.

3.3 Coverage Mechanism

Coverage was adapted for NMT by Tu et al. (2016) and Mi et al. (2016), who both use a GRU to update the coverage vector each move, and is based on Statistical Machine Translation (Koehn, 2009). We discovered that summing the attention distributions to get the coverage vector is sufficient. In this regard, our methodology is close to that of Xu et al. (2015), who use a coverage-like system for image captioning, and Chen et al. (2016), who use a coverage mechanism (distraction) in neural summarization of longer texts.

4. PROPOSED MODEL

In this section we describe (1) our baseline sequence-to-sequence model, (2) our pointer- generator model, and (3) our coverage mechanism that can be added to either of the first two models.

4.1 Sequence-to-sequence attentional model

Figure 1 depicts our baseline model, which is identical to that of Nallapati et al. (2016). The reviews's tokens w_i are fed into the encoder (a single-layer bidirectional LSTM) one by one, resulting

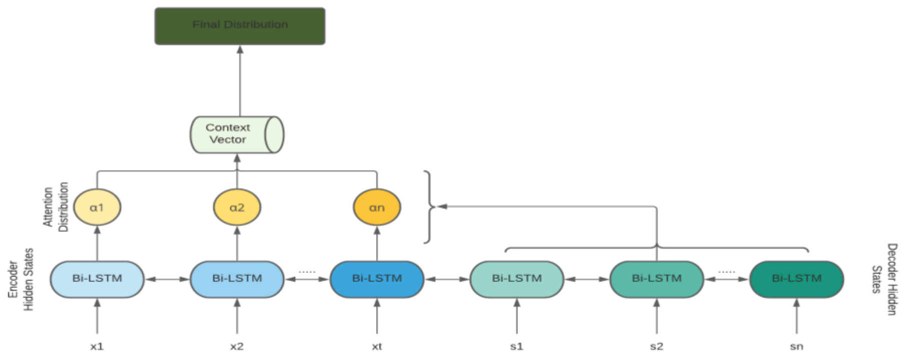

Figure 1. Baseline Sequence-to-sequence model with attention

in a sequence of encoder hidden states h_i. The decoder (a single-layer unidirectional LSTM) has decoder state s_t after receiving the word embedding of the previous word (during preparation, this is the previous word of the reference summary; during testing, it is the previous word emitted by the decoder). According to Bahdanau et al. (2015), the focus distribution is determined as follows:

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + b_{\text{attn}}) \qquad (1)$$
$$a^t = \text{softmax}(e^t) \qquad (2)$$

where v, W_h, W_s and b_{attn} are learnable parameters. The attention distribution can be viewed as a probability distribution over the source words that tells the decoder where to look for the next word to be produced. The attention distribution is then used to generate the background vector, which is a weighted sum of the encoder hidden states, known as context vector h_t^*:

$$h_t^* = \textstyle\sum_i a_i^t h_i \qquad (3)$$

The vocabulary distribution P_{vocab} is created by concatenating the background vector with the decoder state s_t which can be thought of as a fixed-size representation of what has been read from the source for this stage:

$$P_{\text{vocab}} = \text{softmax}(V'(V[s_t, h_t^*] + b) + b') \qquad (4)$$

where V, V', b and b' are learnable parameters. P_{vocab} is a probability distribution over all words in the vocabulary, and provides us with our final distribution from which to predict words w:

$$P(w) = P_{\text{vocab}}(w) \qquad (5)$$

4.2 Pointer-Generator Network

Since it allows both copying words by pointing and generating words from a fixed vocabulary, our pointer-generator network(implemented in figure 2) is a combination of our baseline and a pointer network (Vinyals et al., 2015). The attention distribution at and background vector ht are determined in the pointer-generator model (shown in Figure 2). In addition, the context vector h_t^*, the decoder state s_t and the decoder input x_t are used to measure the generation probability $p_{\text{gen}} \in [0, 1]$ for timestep t:

$$p_{\text{gen}} = \sigma(w_{h^*}^T h_t^* + w_s^T s_t + w_x^T x_t + b_{\text{ptr}}) \qquad (6)$$

where the learnable parameters are the vectors w_{h^*}, w_s, w_x and scalar b_{ptr}, and σ is the sigmoid function. Then, using p_{gen} as a soft switch, you can choose between producing a word from the vocabulary by sampling from P_{vocab} and copying a word from the input sequence by sampling from the attention distribution a^t. Allow the expanded vocabulary to signify the union of the vocabulary and all words appearing in the source document for each document. Over the expanded vocabulary, we obtain the following probability distribution:

$$P(w) = p_{\text{gen}} P_{\text{vocab}}(w) + (1 - p_{\text{gen}}) \textstyle\sum_{i:w_i=w} a_i^t \qquad (7)$$

4.3 Coverage Mechanism

Repetition is a common issue with sequence-to-sequence models and it is exacerbated when creating multi-sentence text. To solve the problem, we use Tu et al., **(**2016**)** coverage model. We keep a coverage vector c^t in our coverage model, which is the number of all previous decoder timesteps' attention distributions:

$$c^t = \textstyle\sum_{t'=0}^{t-1} a^{t'} \qquad (8)$$

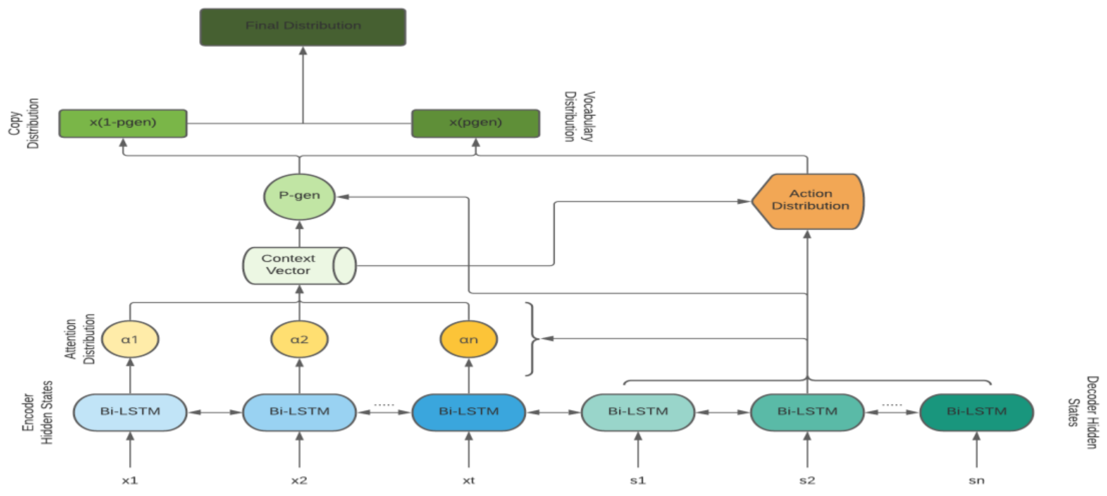

Figure 2. Our pointer-generator model

c^t is a distribution over the source document terms that reflects the degree of publicity such words have obtained so far from the attention system. Since no part of the source document has been protected on the first timestep, c^0 is a zero vector. The attention mechanism is given an extra input in the form of the coverage vector, changing equation (1) to:

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + w_c c_i^t + b_{\text{attn}}) \qquad (9)$$

where w_c is the same length as v and is a learnable parameter variable. This means that the focus mechanism's current decision is told by a reminder of previous decisions. This should make it easier for the attention system to stop attending to the same locations repeatedly, avoiding the generation of repetitive text.

4.4 Word Embedding

The baseline code depended on word2vector embedding, however, another word embedding approach is fastText, which is an extension of the word2vec model. FastText portrays each word as an n-gram of characters rather than learning vectors for words directly. Even if a word was not seen during training, its embeddings can be determined by breaking it down into n-grams. FastText performs better than Word2Vec and allows uncommon terms to be expressed correctly, while taking longer to train (number of n-grams > number of words).

5. EXPERIMENTS

5.1 Dataset

We used the amazon fine food review dataset, which is about review of fine foods from amazon. It is around 642 MB of data with more than 586 thousand reviews.

5.2 Experimental Settings

We changed the algorithm from greedy algorithm into beam search to reduce the computational power during training. Moreover we used GPU due to ram utilization as we faced a problem loading the entire dataset into ram. That is why we only used 50k of the dataset and added early stopping.

5.3 Hyperparameters Experiments

Table 1. Hyperparameters experiment results

PARAMETER	EXP 1	EXP 2	EXP 3	EXP 4
DROPOUT	0.4	0.4	0.4	0.3
ACTIVATION	softmax	softmax	softmax	softmax
L1	Le-5	Le-5	Le-2	Le-2
L2	Le-4	Le-4	Le-2	Le-2
OPTIMIZER	RMSProp	Adam	Adam	RMSProp
LOSS FUNCTION	Sparse categorical cross entropy	Sparse categorical cross entropy	Sparse categorical cross entropy	Sparse categorical cross entropy
LSTM LAYER	1	2	3	5
LOSS	2.79	1.99	1.6	1.2
ACCURACY	0.43	0.57	0.709	0.745

Comparing the most 4 effective experiments that we had as seen in Table 1, we have found that the activation function that mostly fits is softmax. And that we need to use both regularizers L1 and L2, While the best match for the optimizers is RMSprop. Also, the most suitable loss function is Sparse categorical cross entropy. Moreover, we found that the optimal number of our stacked LSTM layers is 5. And finally we have achieved a good loss results of 1.2 with a percentage accuracy around 75% as seen in Figure 3 and 4.

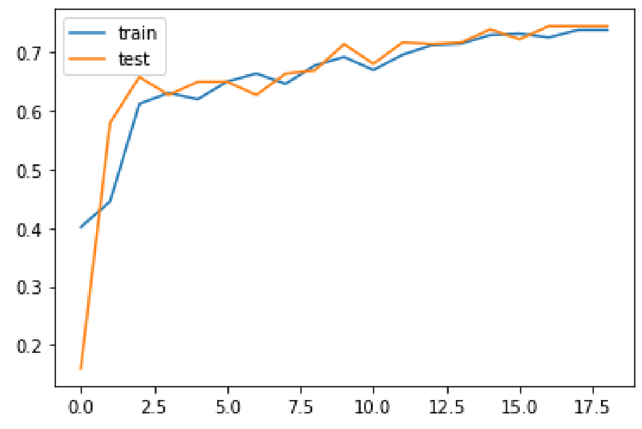

Figure 3. Model achieved accuracy of 75% compared to 72% of baseline code.

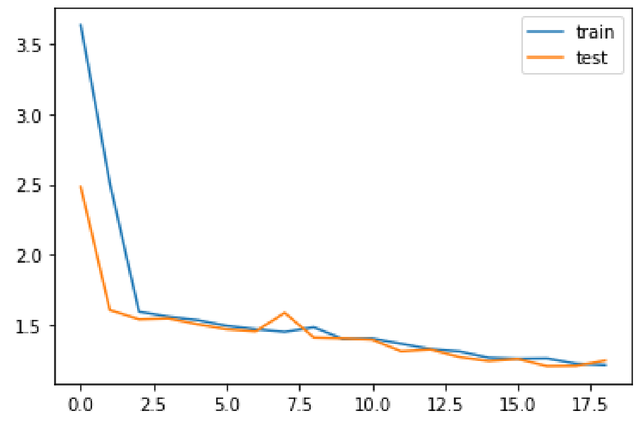

Figure 4. Model achieved loss of 1.2 compared to 1.4 of baseline code.

5.4 Evaluation Metrics

Comparing the results of our model with state-of the art models we used ROUGE-1 which refers to overlap of unigrams between the system summary and reference summary, we also used ROUGE-2 which refers to the overlap of bigrams between the system and the reference summaries. And finally ROUGE-L which measures the longest matching sequence of words. We also measured BLEU performance which evaluate the quality of machine summarized text. All of which showed that our final model had much better and higher quality results than all previous state of arts as seen in Table 2 below.

Table 2. Evaluation Metrics results

MODEL	ROUGE 1	ROUGE 2	ROUGE L	BLEU
ABSTRACTIVE MODEL	35.46	13.30	32.65	-
SEQ-TO-SEQ + ATTN BASELINE (150K VOCAB)	30.49	11.17	28.08	-
SEQ-TO-SEQ + ATTN BASELINE (150K VOCAB)	31.33	11.81	33.42	-
POINTER-GENERATOR	36.44	15.66	33.42	-
OUR MODEL	43.95	22.22	40.14	78.67

5.5 Sample Outputs

In Table 3 is a sample of some of the outputs that we have got which contain people's reviews on some of the food outlets alongside the original summary, base model summary and our model summary.

Table 3. Output Samples

Sentence: wonderful flavor would purchase this blend of coffee again light flavor not bitter at all and price was great the best i found anywhere **Our model Summary:** great coffee flavor **Baseline Summary:** good flavor **Actual Summary:** wolfgang puck k cup breakfast in bed.
Sentence: the pepper plant habanero extra hot california style hot pepper sauce 10 oz has great flavor as all the pepper plants do i just love it it is a bit pricey but worth it **Our model Summary:** great seasoning **Baseline Summary:** great flavour **Actual Summary:** wonderful love it
Sentence: once more amazon was great the product is good for kids even though it has a little bit more sugar than needed **Our model Summary:** good as expected **Baseline Summary:** good **Actual Summary:** as expected

6. CONCLUSION

So in summary, we introduced a hybrid model with a pointer-generator method and a coverage mechanism to overcome the obstacle of long text summarization, which is also an extension and convergence of state-of-the-art models. The method combines the benefits of both extractive and abstractive approaches, outperforming other methods in terms of accuracy and convergence speed. In long text summarization, our model can also manage the problem of running out of vocabulary and the problem of repetition.

7. FUTURE WORKS

We plan to continue our research on the following aspects: although this model performs well, and with just a subset of the data, it would be neat to expand the architecture to improve the quality of the generated summaries. Also, we will try to take the model one step further and apply it on a web or mobile application.

8. ACKNOWLEDGMENTS

We would like to thank the effort and support presented by both; the course Instructor Dr. Mostafa Youssef and Graduate TA Khaled El-Tahan

9. REFERENCES

[1] Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In Empirical Methods in Natural Language Processing.

[2] Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In North American Chapter of the Association for Computational Linguistics.

[3] Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In Empirical Methods in Natural Language Processing.

[4] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In Association for the Advancement of Artificial Intelligence.

[5] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In Computational Natural Language Learning.

[6] Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In Empirical Methods in Natural Language Processing.

[7] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In International Conference on Learning Representations.

[8] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In Neural Information Processing Systems.

[9] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In Neural Information Processing Systems.

[10] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In International Conference on Learning Representations.

[11] Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In Association for Computational Linguistics.

[12] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. In NIPS 2016 Workshop on Multi-class and Multi-label Learning in Extremely Large Label Spaces.

[13] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Techni- cal report, OpenAI.

[14] Chin-Yew Lin. 2004. ROUGE: A package for auto- matic evaluation of summaries. In Text Summariza- tion Branches Out, pages 74–81, Barcelona, Spain.

[15] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018a. Don't give me the details, just the summary! topic- aware convolutional neural networks for ex- treme summarization. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1797–1807, Brussels, Belgium.

[16] Ilya Sutskever, Oriol Vinyals, and Quoc Le. 2014. Se- quence to sequence learning with neural networks. In NIPS. Vishal Gupta and Gurpreet Singh Lehal. 2010. A survey of text summarization extractive techniques. Journal of

[17] Emerging Technologies in Web Intelligence, 2(3):258–268.

[18] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between hu- man and machine translation. In arXiv preprint arXiv:1609.08144.