THE AMERICAN UNIVERSITY IN CAIRO
الجامعة الأمريكية بالقاهرة

Text Summarization with Pointer-Generator Networks

Dareen Hussein, Israa Fahmy

PMDL Course – AUC CSCE Department – 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. 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.

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.

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.

Base Model

The base model paper was about an Encoder-decoder recurrent neural network with long short-term memory (LSTM) units and attention to generate summarizes for food reviews. We found that this model is quite effective abstractive summarizer that generate new sentences that might not be present in the original text.

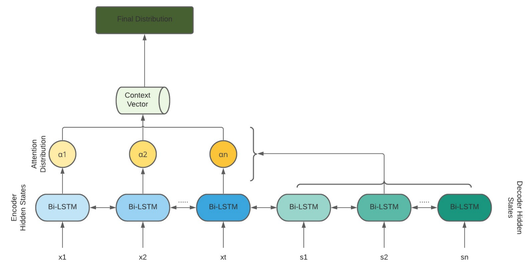

Figure 1. Baseline Sequence-to-sequence model with attention

It uses both training and holdout loss, and uses an embedding layer that transforms the word into a vector. Moreover it depends on ROUGE evaluation metrics for abstractive performance testing. Whereas the baseline architecture (as seen in Fig1), depended on an embedding wor2vector that fed the encoder then output into the decoder with a third party attention mechanism that implements a greedy search algorithm.

Solution Model

So our proposed solution to the baseline model was to make it; an abstractive summarizer which generate new sentences from original ones, uses a seq2seq encoder decoder architecture with bi-directional LSTM, applies a customized attention mechanism which aims to predict a word by looking at a few specific parts of the sequence only, and finally uses a beam search strategy instead of greedy approach to reduce the computational power.
However, the Seq2Seq architecture still suffers from 2 issues. The generated summaries may produce factually incorrect information and can contain many repeated words. So our proposed solution was as follows:

I. Pointer-Generator Network

Since it allows both copying words by pointing and generating words from a fixed vocabulary, our pointer-generator network (implemented in Fig2) is a combination of our baseline and a pointer network At each decoder timestep a generation probability is calculated, which determines whether to generate words from the vocabulary or copy words from the source. The vocabulary and the attention distribution are weighted and summed at the end to form the prediction.

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

So you just apply transformations to everything that you have and then sigmoid to get probability. Then the training objective for our model would be cross-entropy loss with this final distribution.

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

II. Coverage Mechanism

We then build on top of our point-generator model with a coverage, this coverage vector will know that certain pieces have been attended already many times so it sums attention distributions over all previous decoders and introduce an extra loss term to minimize the minimum of attention probabilities. This penalizes the network for attending to the same words again.

$$\textbf{Coverage vector} \quad c^t = \sum_{t'=0}^{t-1} a^{t'}$$

The attention mechanism is given an extra input in the form of the coverage vector, This should make it easier for the attention system to stop attending to the same locations repeatedly, avoiding the generation of repetitive text.

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

III. FastText 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).

So our final model architecture ended up as follows

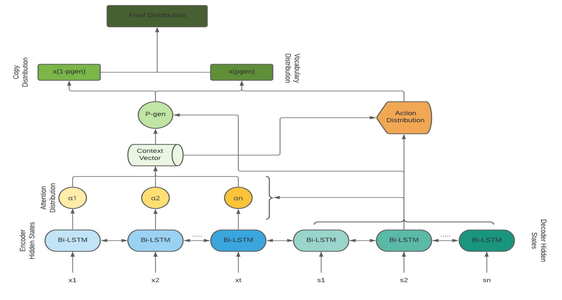

Figure 2. Our pointer-generator Model

Results

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.
- We need to use both regularizers L1 and L2.
- The best match for the optimizers is RMSprop.
- Most suitable loss function is Sparse categorical cross entropy.
- Optimal number of our stacked LSTM layers is 5.

Table 1. Hyperparameters Experiments Results

Parameter	Experiment 1	Experiment 2	Experiment 3	Experiment 4
Dropout	0.4	0.4	0.4	0.3
Activation	softmax	softmax	softmax	softmax
L1	1e-5	1e-5	1e-2	1e-2
L2	1e-4	1e-4	1e-2	1e-2
Optimizer	RMSprop	Adam	Adam	RMSprop
Loss function	Sparse categorical cross entropy	Sparse categorical cross entropy	Sparse categorical cross entropy	Sparse categorical cross entropy
Stacked LSTM layers	1	3	3	5
Loss	2.79	1.99	1.6	1.2
Accuray	0.43	0.57	0.7.09	0.745

All of which led us to achieve a good loss results of 1.2 (Fig3) with a percentage accuracy around 75% (Fig4) compared to 1.4 loss and 72% accuracy in the base model.

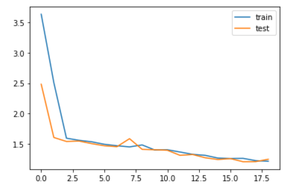

Figure 3. Our Model Loss

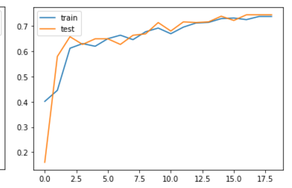

Figure 4. Our Model Accuracy

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.

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 (50k vocab)	31.33	11.81	28.83	-
pointer-generator	36.44	15.66	33.42	-
Our Solution	43.95	22.22	40.14	78.67

Below 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.

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 flavour
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

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.

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.

Acknowledgements

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

References

Scan this QR Code to get a list with the referenced papers and projects.