Using Encoder-Decoder Model to Summarize Customer Reviews on Amazon

Version 1.0

Gang Yang

Abstract

Text summarization is a challenging and interesting topic in the Natural Language Process. There are two types of summarization, one is abstractive, the other is extractive. The main idea is generating summary from a given text. In this paper, I focus on using the encoder-decoder model with attention mechanism to summarize the Amazon customer reviews. To show the model's performance, I use ROUGE as the metrics to compare the referenced summary and generated summary. Comparing to the state-of-art models of text summarization, the model shown in this paper does not perform ideally and I would discuss more details later.

1 Introduction

In order to evaluate products' quality and popularity, most e-commerce platforms, like Amazon, Yelp, Alibaba, etc, would recommend customers to write reviews after buying products. These reviews are valuable for both merchants and potential buyers. For merchants, after reading the reviews, they would know which aspects their products should improve; for potential buyers, they would realize whether that product is worthwhile to buy. However, not all the reviews are brief and straightforward, which would not easy for people to evaluate the goodness of products. Thus, it is necessary to build some models to summarize the reviews on e-commerce platform to help customers make decisions more quickly, and let merchants realize the advantages and disadvantages of their products. Furthermore, summarizing reviews is also beneficial for the platform itself. Once customers improve their purchase efficiency and merchants provide better products, the platform's earnings and reputation would both increase. In this paper, I use the customer reviews dataset from e-commerce platform Amazon, and apply the Encoder-Decoder model with the Attention mechanism to summarize the reviews. Meanwhile, besides the Long-Short Term Memory layer, I use the Gated Recurrent Unit in my model, which could decrease the model parameters and improve the running speed. Finally, I use ROUGE as the metrics to testify the model performance. It could show how similar the referenced summary and the generated summary. I build different structures in my encoder-decoder model and the best one has ROUGE-L score in 21.61, which is not ideal but acceptable. For the details about the model and metrics, I would discuss later.

Since text summarization is a hot topic in the Natural Language Process, many scientists build the models with the outstanding performance related to this topic. In the article *ProphetNet: Predicting Future N-gram for Sequence-to-Sequence Pre-training*, Yu Yan, Weizhen Qi and six other scientists (Yan et al., 2020) use ProphetNet, a new Seq2Seq pre-training model, to summarize the CNN/Daily Mail dataset, and get ROUGE-L score in 40.72. Of course, there are some limitations existed in the research of text summarization. In the article *Neural Text Summarization: A Critical Evaluation*, Caiming Xiong, Richard Socher and three other authors (Kryscinski et al., 2019) discuss the stagnation of text summarization performance from three aspects, datasets, evaluation protocol and the limitation from models. First, they mention some dataset have the situation of layout bias. Also, they demonstrate that the ROUGE metrics has weak correlation with human judgement.

Besides simplifying the customer reviews, text summarization can be applied to many areas. For example, in the financial research area, once the market reports are summarized, business analysts would avoid spending lots of time reading the documents, which let them capture the market behavior more quickly. Also, for the journalists, they could apply this model on the news and generate the title without reading the whole news.

2 Problem Definition and Data

My motivation to do customer review summarization is because online shopping is very normal for contemporary people and reading customer reviews is one important step before buying something. If there is a model could highly summarize customer reviews, it would be applied on lots of e-commerce platform. Also, to testify whether my model could summarize the customer reviews well, using the dataset from Amazon is an appropriate choice. Furthermore, to evaluate the model performance, I would use two different approaches: one is from the quantity, the other is from the quality. For the quantity evaluation, I would apply ROUGE to show whether my predicted summary is close to the referenced summary (the ground truth); for the quality evaluation, I would randomly choose some predicted summary, and compare with the man-made summary to see whether they have the similar meaning semantically. For the latter one, I would judge it by my own.
The dataset is downloaded from `http://jmcauley.ucsd.edu/data/amazon/`, and I will use the customer reviews of home&kitchen products. There are totally 551682 instances. For each instance, there are 9 features and I will mainly use ”reviewText” and ”summary”. The ”reviewText” is the original customer reviews without modification; the ”summary” is the corresponding review summarization generated by data providers. Here, I would show two instances of the dataset. For one instance, the ”reviewText” is ”Nice quality and hold up well in multiple washings? I would buy these again and the price was very good” and the corresponding ”summary” is ”Good quality”; another instance, the ”reviewText” is ”Got this for my girlfriend who runs a baking business and she said it was handy to have for a reference book. She said it is handy to have when thinking of decorating ideas.” and the ”summary” is ”Handy to Have”. From these two arbitrary examples, we find the man-made ”summary” highly summarize the ”reviewText”. Generally speaking, although the raw data is fairly clean, I still need to use some methods to preprocess the data, like expanding the abbreviation, removing the stop words, punctuation and so on. After preprocessing the data, I would like to check the distribution of the text length for both reviews and summary, which could give me the general idea what’s the length range for the majority of the data. And this information could help me set the input length for my encoder-decoder model. Finally, I find 95% reviews have length less than 154 and 95% summaries have length less than 6. So, I only pick the data with both review’s length less than or equal to 154 and summary’s length less than or equal to 6.

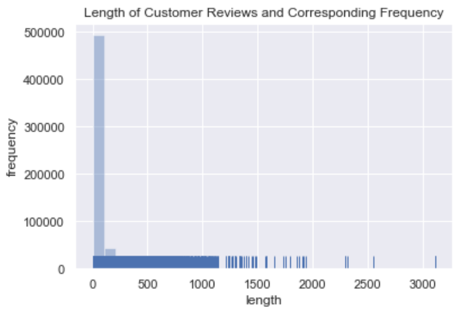


Figure 1: The distribution of the text length for customer reviews

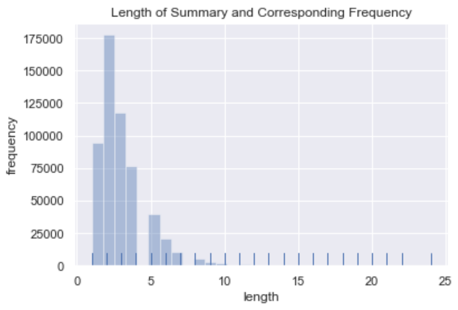


Figure 2: The distribution of the text length for summaries

3 Related Work

Text summarization is a quite interesting and challenging topic in natural language process. In the past few years, many researchers and scientists studied this topic and made extraordinary achievements. In the article *Abstractive Summarization for Amazon Reviews*, Lu Yang (Yang, 2016) applied three models to do the abstractive summarization on the customer reviews. These models include the feed-forward NN with Attention-based Encoder, RNN Encoder-decoder with Attention Mechanism and Attentive RNN. In his experiment, although the first model has the higher perplexity than the second and third model, its summarization result looks better than the other two. But after doing more research on abstractive summarization, I believe the reason that RNN-based model does not perform well in this paper is Lu did not implement RNN model well, and he admitted this point in his paper. In the article *Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond*, Ramesh Nallapati, Bowen Zhou and the other three researchers (Nallapati et al., 2016) mainly used Attentional Encoder-Decoder RNN to do the text summarization. And they improve the model by capturing keywords using feature-rich encoder, modeling rare/unseen words using switching generator-pointer and capturing hierarchical document structure with hierarchical attention. The results from the experiments outperform the state-of-the-art results based on the same dataset. Furthermore, since many text summarization papers use ROUGE as the evaluation metric, I dig more about this metrics. *Looking for a Few Good Metrics: ROUGE and its Evaluation* is a good paper to introduce ROUGE. Chin-Yew Lin (Lin and Och, 2004) discusses five different ROUGE metrics. And I mainly apply ROUGE-1, ROUGE-2 and ROUGE-L to evaluate my model's performance. Of course, there are some voice criticizing the current approaches to do the text summarization. In the article *Neural Text Summarization: A Critical Evaluation*, Caiming Xiong, Richard Socher and three other authors (Kryscinski et al., 2019) discuss the stagnation of text summarization performance from three aspects, datasets, evaluation protocol and the limitation from models. After reading this paper, I become more cautious when doing my own project. First, they mention some dataset have the situation of layout bias. So, I choose to use the customer reviews as the dataset. Also, authors demonstrate that the ROUGE metrics has weak correlation with human judgement. So, besides using ROUGE metrics as the quantity evaluation, I add the quality evaluation to compare the predicted summarization and the reference summarization.

To sum up, using encoder-decoder model to solve the text summarization problem is the mainstream nowadays. I also use this kind of model in this paper. But, I use different layers to construct neural network in the encoder and decoder parts. Although my model's performance is not as good as the models' performance mentioned in the above papers, I get deeper insights in text summarization, which could let me do better in the future work.

4 Methodology

4.1 Data Extraction & Preprocess

Since the download file has JSON format, I first transform it to the txt file. Then, I extract the "reviewText" and "summary" parts from each instance. Once I extract the features, I start to clean the text, both customer reviews and referenced summaries. Since there are lot's of "it's", "I'm", "they're" in the text, I should expand these abbreviation. Then, I set all the words to the lower case. Also, I remove the punctuation, stop words and so on. After I get the clean data, I want to check the length distribution of my dataset in order to determine the size of input. I find 95% customer reviews have length less than or equal to 154, and 95% generated summaries have length less than or equal to 6. Also, I add the "start" token and the "end" token at the beginning and end of the summary. The start token will notify the decoder to start decoding, and the end token will let the decoder stop generating the tokens and avoids producing useless tokens. The reason I did not do the same thing on the customer reviews is I fix the input length for each instance. After that, I use the text preprocessing module in TensorFlow[1]. Tokenizer is the class under preprocessing module, which could help us to transform the text sequence to the integer sequence. To set the number of words we want to use for our tokenizer, I need to count the number of the unique words and the number of the rare words in our dataset. And I set 5 as the threshold, which means if the number of the word appearance is less than 5, I regard it as the rare word. Meanwhile, I use the pad_sequences class to supplement 0 at the end of the integer sequence if the length is less than 154. I did the same thing on the generated summary. By now, we have the input we want for our encoder-decoder model.

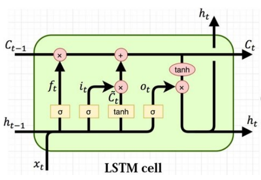
4.2 Model

4.2.1 Encoder-Decoder Model with One LSTM Layer & Attention Mechanism

The encoder-decoder model is the model mainly using recurrent neural network or its variant to do the sequence-to-sequence prediction problems. I mainly use the TensorFlow[2] to construct

[1] https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing
[2] https://www.tensorflow.org/api_docs/python/tf/keras/layers

the Encoder-Decoder model, and refer the code framework on the website[3]. The first part is the encoder. I use the Input layer with input shape equals to 154, and then add the Embedding layers to transform my integer sequence to dense vectors. Since the data has sequence format and I want to capture the internal relationship for each data at different timestamp, LSTM layer is a good option here.



$$i_t = \sigma(x_t U^i + h_{t-1} W^i)$$
$$f_t = \sigma(x_t U^f + h_{t-1} W^f)$$
$$o_t = \sigma(x_t U^o + h_{t-1} W^o)$$
$$\tilde{C}_t = \tanh(x_t U^g + h_{t-1} W^g)$$
$$C_t = \sigma(f_t * C_{t-1} + i_t * \tilde{C}_t)$$
$$h_t = \tanh(C_t) * o_t$$

Figure 3: Long Short-Term Memory

Then, I construct the decoder part, which is similar to the encoder part. The decoder includes the Input, Embedding and LSTM layers. Meanwhile, I use the output generated from the encoder LSTM layer as the input for the decoder LSTM layer.

After building the encoder and decoder part, I introduce the Attention mechanism. As we all know, the traditional encoder-decoder model has some drawbacks. For example, when transferring information from encoder to decoder, it will lose some detailed information since the decoder is only based on one semantic vector and the source sequence will be compressed in a size-fixed sequence. The attention mechanism will solve this problem by letting the decoder go back to check the hidden state of the source sequence. The formula below shows the core idea about the attention mechanism. h_i is the hidden state for the encoder. α_{ij} is the attention weight, which could capture the relationship between the encoder hidden state and the decoder hidden state. After combining the attention weight and the encoder hidden state, we could get the new vector c_j and introduce it to the decoder.

$$c_j = \sum_{i=1}^{T} \alpha_{ij} h_i \tag{1}$$

I use the third-party implementation on GitHub[4] to build the attention layer.

[3] https://machinelearningmastery.com/encoder-decoder-models-text-summarization-keras/

[4] https://github.com/thushv89/attention_keras/blob/master/layers/attention.py

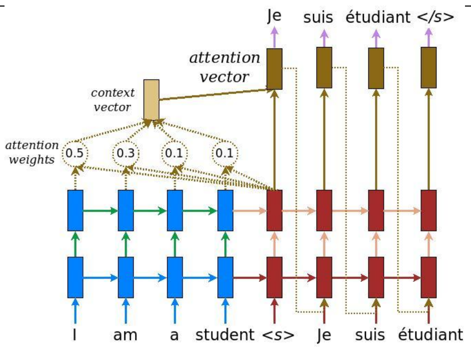


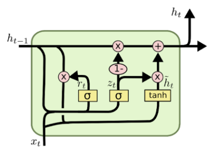
Figure 4: Attention Mechanism

4.2.2 Encoder-Decoder Model with Three LSTM Layer & Attention Mechanism

For the Encoder-Decoder model with three LSTM layer, I use almost the same structure as the model with one LSTM layer. Instead one LSTM layer in the encoder part, I use three LSTM layers. The reason I want to try this model is I think more complicated neural network might capture more detailed information from the input sequence. I would discuss more comparison in the ”Evaluation and Results” section.

4.2.3 Encoder-Decoder Model with One GRU Layer & Attention Mechanism

For the Encoder-Decoder model with One GRU layer, I use the Gated recurrent unit layer in the encoder part. The GRU has the similar idea with the LSTM but with fewer parameters, since the ”update gate” unites the input and forget gates and the ”reset gate” combines the hidden and cell state. GRU and LSTM are two variant of recurrent neural network, and I want to compare their performance.



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$
$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 5: Gated recurrent unit

4.3 Model Training, Inference Setting and Sequence Decoding

To train the model, I apply ”Adam” as the optimizer. ”Adam” is an adaptive learning rate optimization algorithm. It combines the advantages of both ”AdaGrad” and ”RMSProp”. Meanwhile, I use the ”sparse categorical crossentropy” as my loss function, since there is no ”class” in text summarization problem. In other words, each customer reviews belongs to one class due to the uniqueness of the referenced summary. Then, we set up the inference of the model based on the process of building the model. After that, we create the function used to decode the sequence. The parameter of the function is the preprocessed integer sequence.

5 Evaluation and Results

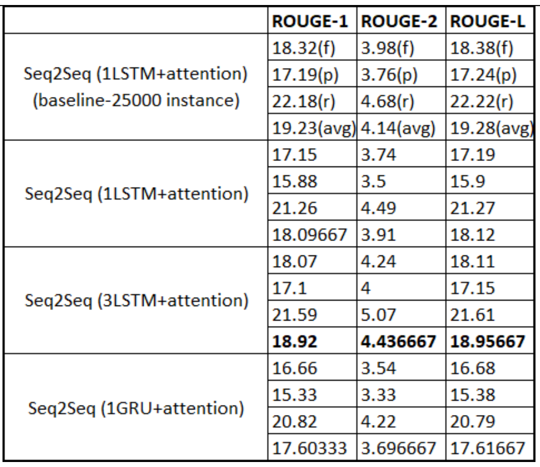
To evaluate the model performance, I mainly use ROUGE-1, ROUGE-2, ROUGE-L as the metrics. For each ROUGE, it has three scores, F1, precision and recall. Precision and recall are easy to interpret. The formula of the recall is

$$recall = \frac{number\ of\ overlapping\ words}{total\ words\ in\ reference\ summary} \tag{2}$$

Based on the above formula, recall is more focus on testing whether the model could capture the information from the referenced summary well. Meanwhile, we do not want to have the tedious summary even if it includes all the information in the referenced summary. So, we need precision to testify the model performance more comprehensively. The formula of the precision is

$$precision = \frac{number\ of\ overlapping\ words}{total\ words\ in\ system\ summary} \tag{3}$$

Thus, precision could check how many portion of the generated summary is relevant.
The table shown below is the result from four models. The first model is the baseline model, which uses 25000 instances to train. The other three models use the full dataset to train. One weird thing from the result is that the baseline model is a little bit better than the other three models. One reason is that although the size of the training data is smaller, the baseline model trains more epochs, which is around 17 times. On the contrary, the other three models train around 8 epochs. And when only comparing the models with full-size dataset, we find the model with three LSTM layers perform the best. One potential reason is that this model has more parameters than the other two, which might capture the information in the text better.



	ROUGE-1	ROUGE-2	ROUGE-L
Seq2Seq (1LSTM+attention) (baseline-25000 instance)	18.32(f)	3.98(f)	18.38(f)
	17.19(p)	3.76(p)	17.24(p)
	22.18(r)	4.68(r)	22.22(r)
	19.23(avg)	4.14(avg)	19.28(avg)
Seq2Seq (1LSTM+attention)	17.15	3.74	17.19
	15.88	3.5	15.9
	21.26	4.49	21.27
	18.09667	3.91	18.12
Seq2Seq (3LSTM+attention)	18.07	4.24	18.11
	17.1	4	17.15
	21.59	5.07	21.61
	18.92	**4.436667**	**18.95667**
Seq2Seq (1GRU+attention)	16.66	3.54	16.68
	15.33	3.33	15.38
	20.82	4.22	20.79
	17.60333	3.696667	17.61667

Figure 6: ROUGE value

Meanwhile, I want to show some generated summaries from the model used to compare with the referenced summaries. I randomly select some instances from the dataset. From the instances below, we find that the model indeed generates the summary that has close meaning to the referenced summary. But when choosing the words, it does not perform ideally.

Review: cook lot know good electric knife roasts makes much easier slice knife great job cut quickly neatly without lot fatigue
Referenced Summary: best electric knife
Generated Summary: great knife

Review: husband 1 cup french press mornings works great small amount need takes little counter space needs
Referenced Summary:works great
Generated Summary: great coffee press

6 Discussion

In Lu Yang's paper, he gets the ROUGE-1 score in 84.01 and ROUGE-2 score in 79.20 for the home and kitchen products reviews from Amazon. So, comparing to his model performance, my model's result is not ideal. My model gets the ROUGE-1 score with around 20, which means it could only predict one word correctly in 5 words. The ROUGE-2 score is much lower, which means the model could not capture the internal relationship in the text sequence. So, there are several things I should do in the future. First, I need to check whether I preprocess the data appropriately. Maybe the data is not clean enough, maybe

I should keep the stop words, and so on. For the model part, the LSTM layer seems not work as expected. So, I should dig more about the encoder-decoder model to make sure I implement the layers correctly. Also, due to the time limitation, I could not try different combinations of parameters.
To sum up, the model's performance is not quite satisfactory. And the models used the full dataset are even worse than the baseline model. Originally, I expected the models used the full dataset could reach 40 of ROUGE-1 score. The reason caused such a difference might be related to the implementation of the encoder part. Also, maybe I could try the pre-trained model, like BERT, in the future.

7 Conclusion

In this paper, I mainly use the encoder-decoder model with attention mechanism to generate the summary from the Amazon customer reviews. The ROUGE-1 score is around 18 and the ROUGE-2 score is around 4. So, comparing to the state-of-art model in text summarization, my model does not perform well. But, it leaves the enough space for improvement. In the future, I could try the pre-trained model, like BERT, to do the word embedding, or adding more complicated neural network in the encoder and decoder parts.

8 Other Things We Tried

I attempt to use the pre-trained model, like BERT, to replace some layers in my encoder parts. But, I meet the compatibility issue when trying to do this operation. For example, the dimension messes up. So, I need to do more research about the BERT to make sure I know how to use it.

9 What You Would Have Done Differently or Next

In the paper *Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond*, authors(Nallapati et al., 2016) use three different approaches to improve the encoder-decoder model, like capturing keywords using feature-rich encoder, or modeling rare words using switching generator-pointer. If I have time, I would like to try these methods. But the prerequisite is I should improve my baseline model's performance.

10 Work Plan

Generally speaking, I almost follow the plan. I preprocess the dataset, build the encoder-decoder model and get the results on time. However, since I am new in deep learning, I do not have time to try different types of neural networks, like CNN. Also, since I am unfamiliar with BERT, I do not have the chance to use the pre-trained encoder.

References

Wojciech Kryscinski, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Neural text summarization: A critical evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. pages 540–551.

Chin-Yew Lin and FJ Och. 2004. Looking for a few good metrics: Rouge and its evaluation. In *Ntcir Workshop*.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023* .

Yu Yan, Weizhen Qi, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. *arXiv preprint arXiv:2001.04063* .

Lu Yang. 2016. Abstractive summarization for amazon reviews.