Abstractive Text Summarization with Encoder-Decoder Models

Deniz Aytemiz

May 6, 2023

1 Abstract

The text summarization is for producing a brief and concise summary that captures the essential information and overall meaning of the original text. The task is done with natural language processing techniques. Although these algorithms can effectively generate summaries, they are limited in their ability to create entirely new sentences and they may also produce grammatical errors. In this work, it is aimed to employ an abstractive text summarization approach for Amazon Fine Food review using a type of transfer learning which the many to many seq2seq model.

2 Introduction

Text summarization in Natural Language Processing can be broadly categorized into two approaches which are, extractive and abstractive summarization. Extractive summarization, aims to identify the key sentences and phrases from the input text and extract those to form a summary whereas abstractive summarization generates a summary of text that is not simply copied from the original text, but instead is a new synthesis of the content that captures the key points and ideas. Abstractive summarization relies on natural language generation techniques to produce summary that is in the author's own words and can be more concise and easier to understand than the original text. For this purpose Amazon Fine Food review is used as dataset that has reviews for food products which are long sequences of text and their corresponding summary reviews which are shorter and conciser reviews.

3 Methodology

In order the tackle the problem of abstractive text summarization, a seq2seq model is built. Its representative structure is shown in Figure 1.

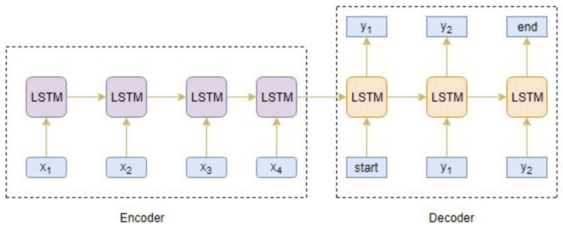
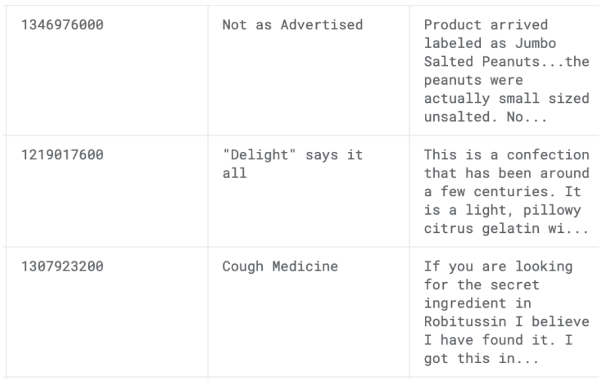


Figure 1: The representative structure of the seq2seq model for abstractive text summarization.

Seq2seq models deal with sequential data for many purposes such as sentiment classification, neural machine translation and named entity recognition. Many to many seq2seq model has multiple inputs and multiple outputs. It has three main parts which are encoder, decoder and attention layer. Encoder network gets the input text data and produces a compact representation of the data which is then fed to decoder network that produces the output sequence. The attention layer, helps the decoder based on its current state to focus on the most relevant parts of the input sequence during the decoding process. This is important because in some cases, different parts of the input sequence may have varying degrees of importance for generating the output sequence.

3.1 Preprocess dataset

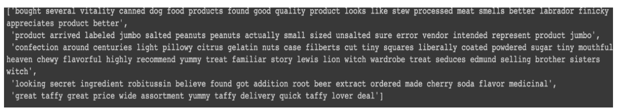
The example of dataset is shown in Figure 2. We extracted first 10000 rows of dataset. The dataset has uneven numbers of summaries and reviews. Hence firstly, we need to clean out the rows that does not have either summary or review. In addition to removing duplicates and NaN values.



1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanuts...the peanuts were actually small sized unsalted. No...
1219017600	"Delight" says it all	This is a confection that has been around a few centuries. It is a light, pillowy citrus gelatin wi...
1307923200	Cough Medicine	If you are looking for the secret ingredient in Robitussin I believe I have found it. I got this in...

Figure 2: Samples from dataset

The review column is preprocessed by converting every word to lowercase, removing HTML tags, removing ('s),removing parentheses, punctuations, special characters, stop words and words that have less than 1 characters. In addition by using a dictionary that maps words such as 'you've' to 'you have' the contractions are replaced with words. The cleaned reviews are shown in Figure 3.



```
['bought several vitality canned dog food products found good quality product looks like stew processed meat smells better labrador finicky
appreciates product better',
 'product arrived labeled jumbo salted peanuts peanuts actually small sized unsalted sure error vendor intended represent product jumbo',
 'confection around centuries light pillowy citrus gelatin nuts case filberts cut tiny squares liberally coated powdered sugar tiny mouthful
heaven chewy flavorful highly recommend yummy treat familiar story lewis lion witch wardrobe treat seduces edmund selling brother sisters
witch',
 'looking secret ingredient robitussin believe found got addition root beer extract ordered made cherry soda flavor medicinal',
 'great taffy great price wide assortment yummy taffy delivery quick taffy lover deal']
```

Figure 3: Cleaned text data

We need to fix the length of both summary and reviews after tokenizing them before inputting them to the model. In order to select which length to fix, first, we need to see the distribution of lengths in both summary and review. It is displayed in Figure 4.

The length to fix text data is chosen to be 30 and summary data chosen to be 8, since they encapsulate most of the samples in both sets. 94 percentage of summary data has length less than and equal to 8, whereas, 54 percentage of text data has length less than and equal to 30. Since the dataset is already large enough, the 54 percentage and 94 percentage of text and summary data will be enough to train and test the model. Therefore, we select the text and summary data that has less than maximum lengths and save it in a new dataframe.

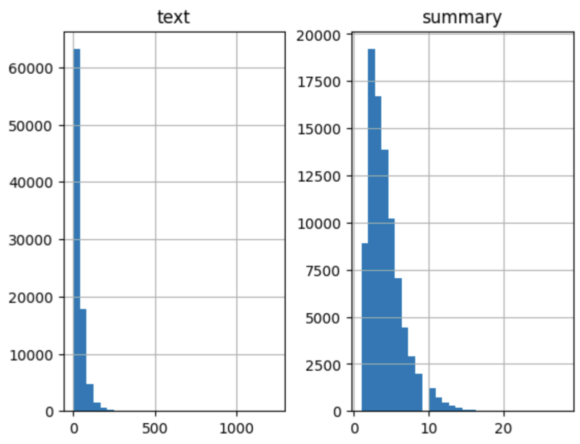
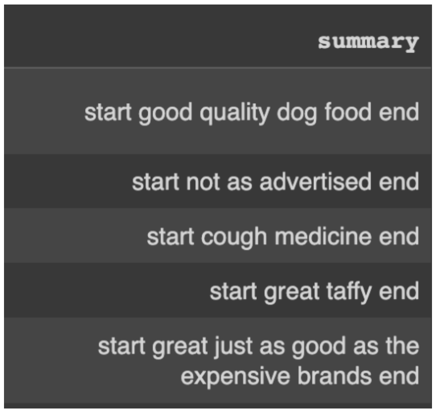


Figure 4: Length distribution in text and summary data

To the summaries in this dataframe, we add start and end tokens as 'start' and 'end'. This is important since they indicate the beginning and end of a sequence, and they help the model learn to generate outputs that are well-formed sentences. In the inference phase, the decoder cannot know when the summary begins or ends. By adding the start and end tokens as ¡start¿ and ¡end¿, respectively, the decoder can use these tokens to determine when to start generating the summary and when to stop.

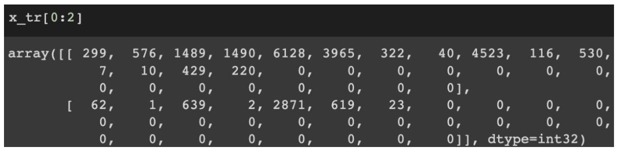
After the dataset is divided to test and training splits, the text and summaries in tokenized in order to find out about the vocabulary of each. Then for each vocabulary of summary and texts training data, the number of rare words are extracted in order to create a more compact vocabulary with the most common words. The number of words that occur less than 4 in text training data is 16487 whereas number of all unique words is 24939 for text training data. We will create another tokenizer object and create a new vocabulary with size 24939-16487 being 8452. The same is done for the summary training data. Hence for both



summary
start good quality dog food end
start not as advertised end
start cough medicine end
start great taffy end
start great just as good as the expensive brands end

Figure 5: Processed Summaries

we have, a more compact vocabulary dictionary. The tokenizer object, converts the text training,test and summary training and test sequences into integers by mapping the index of the keys(words) in the vocabulary dictionary in place of the token in sequences so that the model can process this data. It is similar to word embedding. After these embedded arrays are obtained, each are padded to the maximum number of lengths determined earlier which are 30 and 8 for text and summary respectively. The outcome of the processed training text data is shown in Figure 6. The arrays in summary data that only have 'start'

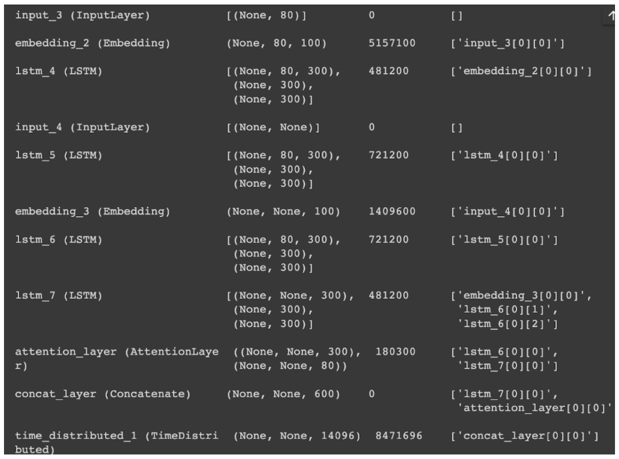


```
x_tr[0:2]

array([[ 299,  576, 1489, 1490, 6128, 3965,  322,   40, 4523,  116,  530,
           7,   10,  429,  220,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0],
       [  62,    1,  639,    2, 2871,  619,   23,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0]], dtype=int32)
```

Figure 6: Processed text data

and 'end' are then removed.

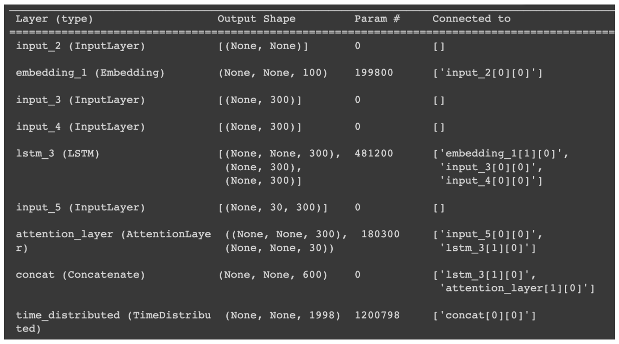
The next step is to construct the attention layer. The attention layer is implemented using Bahdanau attention method which calculates scores for each element in the encoder output sequence and then computes a weighted sum of the encoder output sequence based on these scores to obtain a context vector. This layer gets encoder output and decoder output sequence as inputs. Then the seq2seq model is constructed. The model summary is given in Figure 7.



```
 input_3 (InputLayer)           [(None, 80)]         0           []

 embedding_2 (Embedding)        (None, 80, 100)      5157100     ['input_3[0][0]']

 lstm_4 (LSTM)                  [(None, 80, 300),    481200      ['embedding_2[0][0]']
                                 (None, 300),
                                 (None, 300)]

 input_4 (InputLayer)           [(None, None)]       0           []

 lstm_5 (LSTM)                  [(None, 80, 300),    721200      ['lstm_4[0][0]']
                                 (None, 300),
                                 (None, 300)]

 embedding_3 (Embedding)        (None, None, 100)    1409600     ['input_4[0][0]']

 lstm_6 (LSTM)                  [(None, 80, 300),    721200      ['lstm_5[0][0]']
                                 (None, 300),
                                 (None, 300)]

 lstm_7 (LSTM)                  [(None, None, 300),  481200      ['embedding_3[0][0]',
                                 (None, 300),                     'lstm_6[0][1]',
                                 (None, 300)]                     'lstm_6[0][2]']

 attention_layer (AttentionLaye  ((None, None, 300),  180300     ['lstm_6[0][0]',
 r)                              (None, None, 80))                'lstm_7[0][0]']

 concat_layer (Concatenate)     (None, None, 600)    0           ['lstm_7[0][0]',
                                                                  'attention_layer[0][0]'

 time_distributed_1 (TimeDistri  (None, None, 14096)  8471696    ['concat_layer[0][0]']
 buted)
```

Figure 7: The model summary

In Figure 8, there is the model summary of the inference phase.



```
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================
 input_2 (InputLayer)           [(None, None)]       0           []

 embedding_1 (Embedding)        (None, None, 100)    199800      ['input_2[0][0]']

 input_3 (InputLayer)           [(None, 300)]        0           []

 input_4 (InputLayer)           [(None, 300)]        0           []

 lstm_3 (LSTM)                  [(None, None, 300),  481200      ['embedding_1[1][0]',
                                 (None, 300),                     'input_3[0][0]',
                                 (None, 300)]                     'input_4[0][0]']

 input_5 (InputLayer)           [(None, 30, 300)]    0           []

 attention_layer (AttentionLaye  ((None, None, 300),  180300     ['input_5[0][0]',
 r)                              (None, None, 30))                'lstm_3[1][0]']

 concat (Concatenate)           (None, None, 600)    0           ['lstm_3[1][0]',
                                                                  'attention_layer[1][0]']

 time_distributed (TimeDistribu  (None, None, 1998)  1200798     ['concat[0][0]']
 ted)
```

Figure 8: The model summary of inference phase

4 Experimental Results

In Figure 9, the loss function in each epoch in both train and test sets is displayed. The predictions the model makes are displayed in Figure 10,11,12.

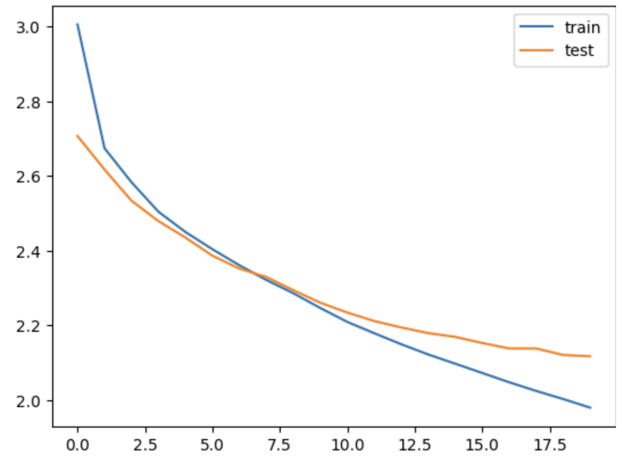
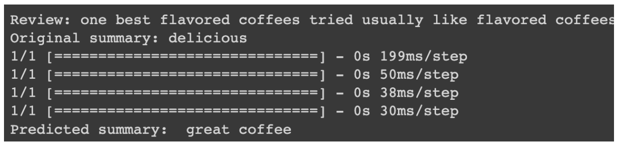


Figure 9: The loss function in each epoch

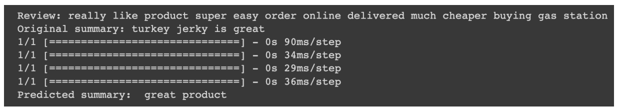


```
Review: one best flavored coffees tried usually like flavored coffees
Original summary: delicious
1/1 [==============================] - 0s 199ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 30ms/step
Predicted summary:  great coffee
```

Figure 10: The predicted summaries

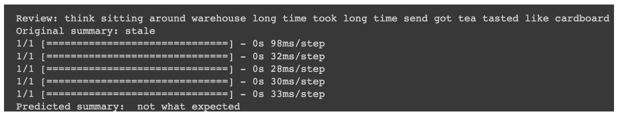
5 Discussion

In Figure 9, since the loss functions in both test and train sets decrease approximately in the same rate, there is no obvious sign of overfitting. Even though the model can predict whether a review is good or bad, there is still room for improvement. Some predicted summaries are repeating such as 'great product'. In addition, even though positive summary is outputted to positive review, some of the predictions are not completly related to reviews and overall can be very general such as instead of 'great cat food', it outputs 'great product'. The overall performance of the model can be improved by expanding the training data and especially training model with more epochs. Using a pre-trained word embedding for English words also may increase the performance. If beam search is added to the model, more diverse and fluent summaries by considering different word orderings and avoiding repeating the same words or phrases can be generated. Since beam search is used in decoding and exploring multiple possible output



```
Review: really like product super easy order online delivered much cheaper buying gas station
Original summary: turkey jerky is great
1/1 [==============================] - 0s 90ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 36ms/step
Predicted summary:  great product
```

Figure 11: The predicted summaries



```
Review: think sitting around warehouse long time took long time send got tea tasted like cardboard
Original summary: stale
1/1 [==============================] - 0s 98ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 33ms/step
Predicted summary:  not what expected
```

Figure 12: The predicted summaries

sequences at the same time. It works by keeping track of a fixed number of the most likely output sequences (called "hypotheses") at each decoding step and then expanding each hypothesis by considering all the possible next words and selecting the top-K most likely ones. This process continues until the end-of-sequence token is generated or a maximum length is reached, and then the hypothesis with the highest score is selected as the final output.

[1]
[2]
[3]

References

[1] J. F. Shaikh, "Essentials of deep learning – sequence to sequence modelling with attention (using python)." Analytics Vidhya, May 2020. 4

[2] V. Saravanan, "Text summarization from scratch using encoder-decoder network with attention in keras." Medium, Jun 2020. 4

[3] A. Pai, "Comprehensive guide to text summarization using deep learning in python." Analytics Vidhya, May 2020. 4