Text Summarization

Authors: Namita Kiran Mahindrakar, Tanmay Alsi, Sakshi Basavanth Patil
Email: mahindrakar.n@northeastern.edu, alsi.t@northeastern.edu, basavanthpatil.s@northeastern.edu

Course: Neural Networks and Deep Learning (IE 7615)

Abstract : Text summarization is the task of producing a concise and fluent summary while preserving the key information and overall meaning of the input data. This project focuses on the abstractive text summarization which paraphrases the document into a concise summary of the original input text. This is done using the sequence-to-sequence (Seq2Seq) Long-Short-Term-Memory (LSTM) model. The model is built with three LSTM iterations, word embedding, encoder-decoder complexity and attention layer. The performance of the model built is evaluated to produce high accuracy comparing the reference summary and the predicted summary of the input text . The dataset used is the Amazon food reviews and CNN daily mail data.

Introduction: The information being generated on a daily basis is a huge amount of textual data. This can be used in the favor of many businesses to increase services, sales or profits just by reviewing the textual information. The problem arises when the reports are huge with time constraints and there is a lack of attention span. Text summarization addresses such issues and is a main application in the natural language programming domain. The current state-of -art model can be achieved by either extractive or abstractive summarization. Extractive approach is comparatively less complex as it involves extracting informative sentences without paraphrasing, whereas in abstractive summarization the summary is generated by deep learning algorithms using LSTM.

The abstractive summary model uses the sequence to sequence model, mapping the input sequence to the target sequence. The sequence to sequence model can be used for process improvement of the state-of-art model to achieve good accuracy on small summary inputs. Similarly attention layer models have been shown to produce high performance evaluation on large summary inputs.

RNN and Seq2Seq (LSTM) Model:

The traditional neural networks do not recall the previous data when building a model. In case of text summarization the model is required to remember the previous output. This is achieved in recurrent neural networks (RNN's) which are looped networks. LSTM is one such RNN used for abstractive summarization using the sequence-to-sequence model. When the input is fed to the encoder decoder seq2seq model, the encoded representation of the model is generated and further fed into the decoder to generate a desired output. Abstractive summarization is comparatively complex as a semantic representation of the document is stored and an accurate vocabulary is selected to fit the semantic to create a summary representing all the key points from the actual input text.

Encoder and Decoder:

Encoder is made up of LSTM units, which overcomes the problem of vanishing gradients. LSTM is known to be efficient for understanding long text sequences, it produces a context vector capturing the entire context and is improvised by using stacked LSTM's. The sentences are converted to integer format based on the term frequency before the encoding process using their frequency and look dictionaries. After the vocabulary, the converted sentence is passed through the decoder for desired outputs.

Decoder input is the encoder output, the previous decoder state output and the previous hidden state. The target word probabilities are calculated by the decoder based on the vector at each time step. The words with the highest probability are selected to produce outputs eventually generating an entire output sequence.

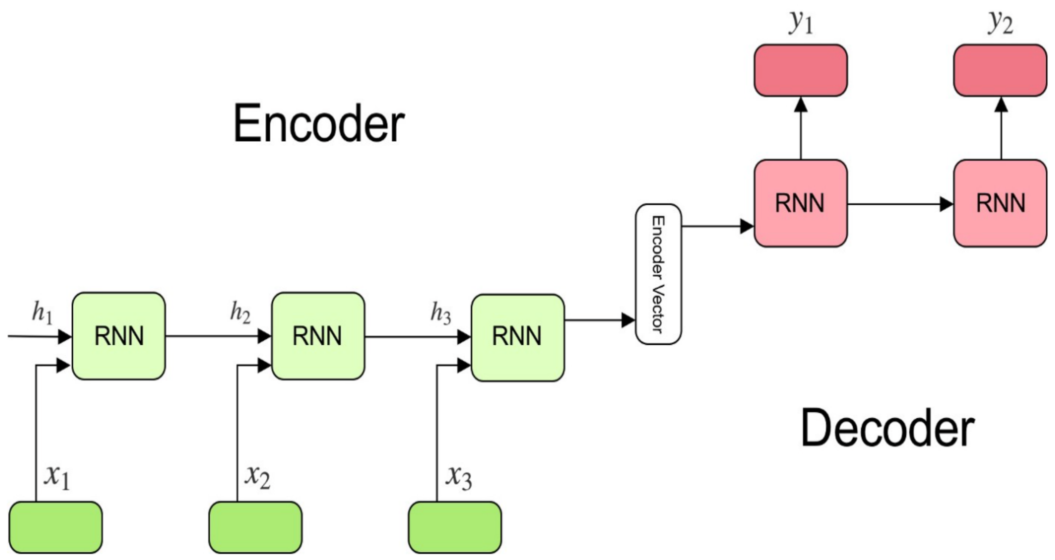

Figure 1: Encoder-decoder Sequence-to-Sequence model

Word Embedding:

Word embedding is an efficient and dense representation of mapping vectors to real numbers building a relationship among words and allows calculations. The higher dimensions of embedding captures fine-grain relationships between the words improving the performance of the model.

Background: Text summarization is implemented using the abstractive approach using the sequence-to-sequence model along with the LSTM model. The abstractive approach is a common sequence-to-sequence deep learning model which is gaining popularity [7]. Seq2Seq models are successfully used in text summarization, speech recognition and machine translation [8]. There have been recent studies on an abstractive approach where the sequence-to-sequence models are using encoders and decoders [9]. Majority of the research papers showed that the encoder-decoder model is a potential solution for text summarization. Research also shows that using LSTM along with encoder decoder models are more efficient in capturing essential information than the traditional RNN. Our project is structured with three LSTM layers in the encoder and a single LSTM layer with the decoder along with an embedded, dense and attention layer.

Approach:

Datasets**:** Amazon food reviews - https://snap.stanford.edu/data/web-FineFoods.html or https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews

The goal of the project is achieved by using the Amazon food review dataset to train the model. The initial step is to filter out and clean the data for null values and any non-alphabetic characters. We need to handle contractions as well as HTML tags.

This text is then tokenized and a model id defined by using the sequence-to-sequence logic by encoding the content of the text (encoder input) and feeding the output as a decoder input to predict the next character in the summarized text.

The model is designed to process the input to produce a concise text summary of the review data. RNNs are used as encoder and decoder. The output vector of the encoder is fed as the input vector to the decoder. This encoder has stacked LSTMs and every hidden state is calculated using the equation:

$$H_t(encoder) = \phi(W_{HH} * H_{t-1} + W_{HX} * X_t)$$

Where ϕ - activation function, H_t (encoder) - hidden state in an encoder, W_{HH} - weight matrix connecting the hidden state, W_{HX} - weight matrix connecting the input and the hidden states.

Similarly the hidden states in the decoder is computed using the equation:

$$H_t(decoder) = \phi(W_{HH} * H_{t-1})$$

The output of the decoder is generated using the equation:

$$Y_t = H_t(decoder) * W_{HY}$$

Where W_{HY} - weight matrix connecting the hidden states with decoder output.

Attention in the encoder decoder neural network allows the context creation at each time step by taking the inputs from the current decoder hidden state and encoder hidden state. The bidirectional LSTM model utilizes different scoring functions to produce context vectors by using the scores as weights in the hidden states. The scoring function uses the softmax function (a) on the raw scores to create a probability distribution across the encoder hidden state to produce context vector using the equation:

$c_t = \sum_s a(s)h$

$a_t (s) = \text{softmax}(s)$ …….(a)

Where c_t - context vector at timestep (t)

The next hidden state in the decoder output is computed using the equation:

$$h_{t+1} = \tanh(W[h_t ; c_t] + b)$$

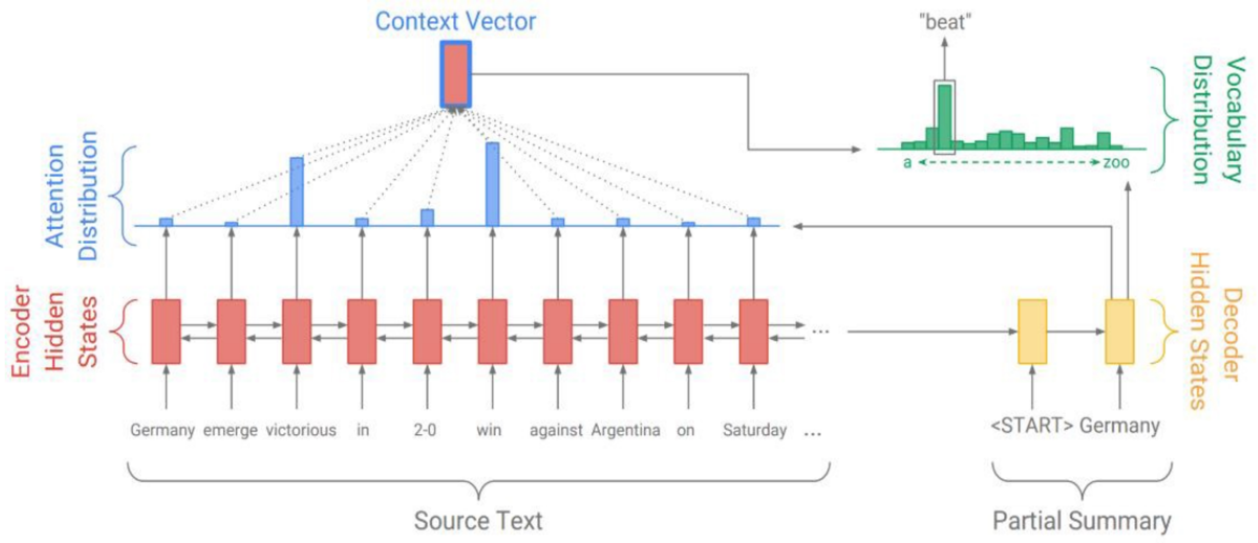

Figure 2: Attention based Encoder-Decoder Network

The abstractive summarization flow diagram is shown in Figure 3 with an example "I have a dog. My dog is tall." The model converts the input into indexes and passes it through the embedded layer to further convert it into vectors. The embedded layer output is passed as the model input to compute the vector matrix of the summary, which is then converted to words using the index to word mapping to produce the output " I have a tall dog."

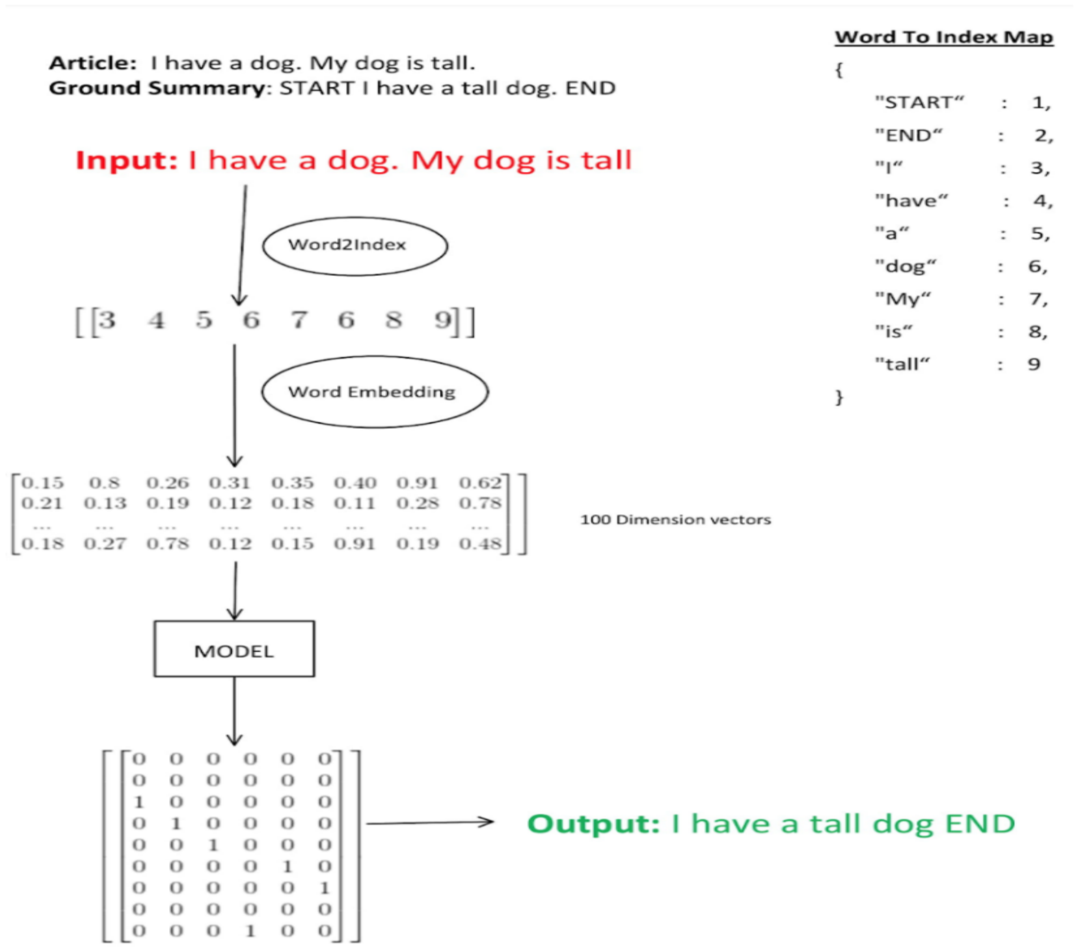
Article: I have a dog. My dog is tall.
Ground Summary: START I have a tall dog. END
Input: I have a dog. My dog is tall
Word2Index
Word Embedding
100 Dimension vectors
MODEL
Output: I have a tall dog END
Word To Index Map
"START" : 1, "END" : 2, "I" : 3, "have" : 4, "a" : 5, "dog" : 6, "My" : 7, "is" : 8, "tall" : 9

Figure 3: Abstractive summarization flow diagram

Result:

Dataset: The dataset we used was Amazon Food reviews which is published by Stanford on their website as well as kaggle.

The data span a period of more than 10 years, including all ~500,000 reviews up to October 2012. Reviews include product and user information, ratings, and a plain text review. It also includes reviews from all other Amazon categories. The two prominent columns we are going to use are Text which has reviews and the summary which has the summaries for the review.

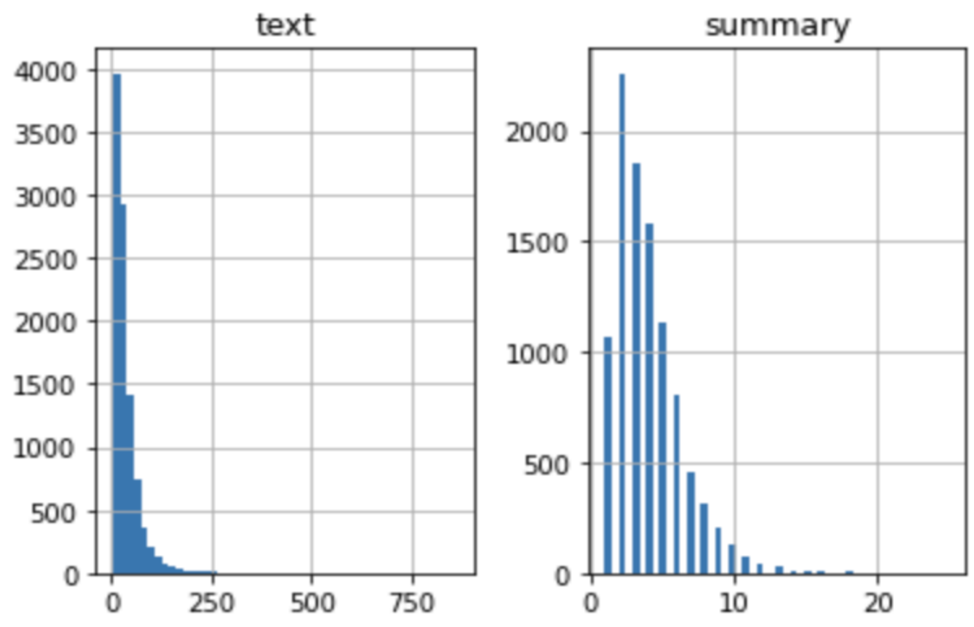

Figure 4: Words distribution in the text and summary data

The above graph shows the distribution of the amount of words in the text and summary column. Using this graph we decided on the maximum length of the reviews and summary to be considered for the training model. Usually this is the mode of the distribution but we took it as 160 and 160 and 20 to cover most of the data.

Experiments and Result:

We trained the model on 200000 rows of the data. This bottleneck on the data was due to lack of available ram on the system. We trained this model with multiple batch sizes and epochs.
Again the main driving factor deciding these hyper parameters were the limited amount of hardware resources at our disposal.
We used a cross-entropy function to calculate the loss and accuracy of the model.

The summaries generated by this model were coherent. But there are times when they either are repeated or just copied from the expected summary.
We Got accuracy of 88% while predicting the summary using multilayer LSTM network (3 Layers)

We also tried experimenting with Bi-directional LSTM and implementing a reverse layer along with a stacked LSTM network. Unfortunately we were unable to get past the memory out of bounds error due to lack of memory.

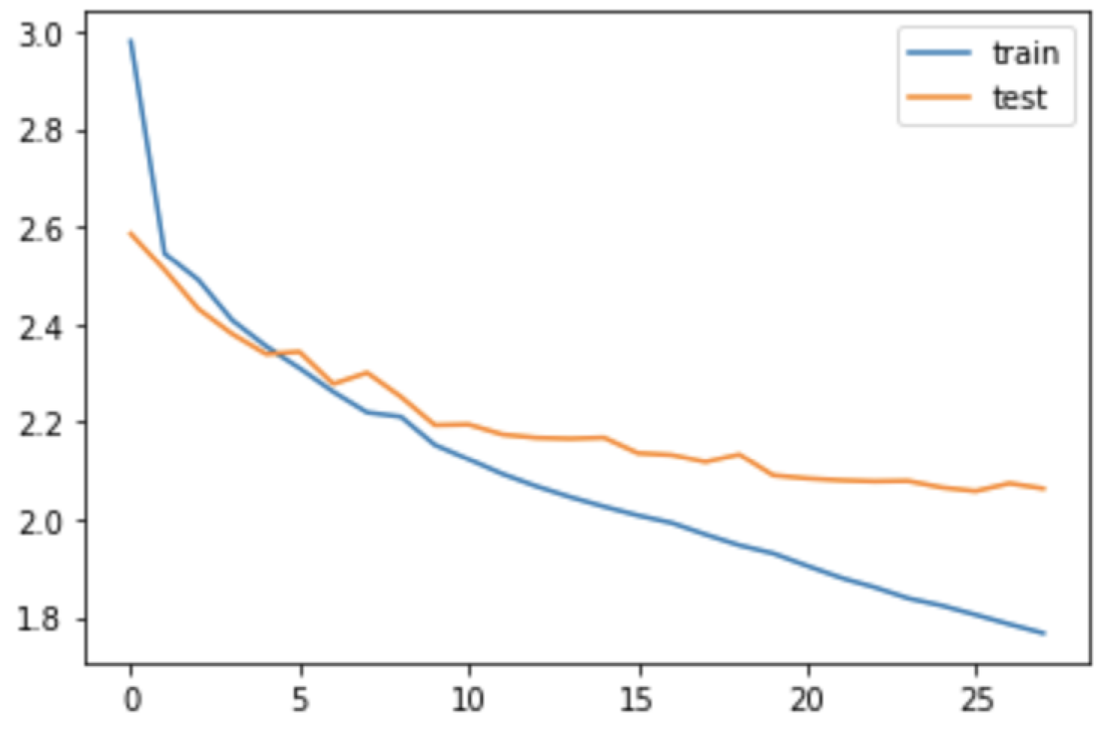

Figure 5: Training and validation loss metrics observed during the training phase for Amazon Food Reviews

Initially we used the concept of early stopping while fitting the model. Essentially it stops the training as soon as the validation loss starts going up. But that is not beneficial every time. As you can see from the above graph the loss function can go significantly lower after the initial spike. Hence, we used the above graph to decide on the number of epochs which came around 24-25 where the loss was either getting steady or rising a little bit.

Below are few results for the reviews taken from amazon.com and their summaries:

Review: I understand many people love these things, but I bought the sea salt and vinegar popchips and I hated them. I LOVE regular potato chips lay's and other brands. These popchips not only taste bad but they have a bad after taste to me.<br />I think these popchips must be for "heath" type people who try to convince themselves these taste good.<br />We like what we like and we all have different experiences about this type of thing.<br />After reading so many positive reviews I thought these would be good.<br />They taste "hallow", "empty", too artificial chemical taste.<br />With a regular potato chip I get the "that's it" sensation, satisfying, full flavor.

Predicted summary: tangy snack

Review: I love chips...and have been noshing on the regular Popchips for months now...love most of the flavors. So when I saw the multigrain, I thought, why not? UGH...it wasn't crispy and the taste was bland and dull. I am so disheartened to have 23 bags left, that I don't know if I will continue my regular subscription. Do not get these unless you want chips that would be better served as packing material

Predicted summary: worst ever tasted

Review: This is one of my favorite kinds of coffee for my Keurig. It has a nice dark taste that is not overpowering while still being strong. My mother hates strong coffee and when she comes to my house she will resuse one of these K-Cups after I've made mine to knock the punch out. Whatever your pleasure, I guess!

Predicted summary: strong coffee

Review: This is probably the best coffee made for the Keurig single serve brewer. Bold statement, I know, but this is one bold coffee that stands apart from the mediocre "Green Mountain" blends that Keurig offers. This is a strong, in your face, wake you up, no holds barred coffee that I wish they made available in whole bean bags (or even pre-ground, it's that good) just so I could take it with me for when I didn't have my Keurig. The only downside is that it can get a little expensive, especially with how addictive it is to drink.<br /><br />I would NOT recommend this coffee for anybody that like to flower up their morning brew with sugar, cream, sweeteners, or heaven forbid those awful flavored creamers.<br /><br />If you like good coffee, drink this and you will not be disappointed.<br /><br />Written while sipping a cup of Black Tiger coffee.

Predicted summary: wow strong coffee good

Review: I gave these a try for the great price. Unfortunately in this case, you get what you pay for. Other more popular well known Keurig brand is MUCH better.

Predicted summary: great value money

Review: I'm not a picky eater but this is by far the worst tasting thing I have every had (and I've had Moxie). It has a bitter taste almost like warm pennies and the flavor stays in your mouth. I couldn't even finish the cup and don't plan on using any of the others. Not only does it taste bad but the smell itself as bad as the flavor. I would not recommend this product!

Predicted summary: disgusting

Discussion: The main limiting factor we faced during our experiment was the lack of hardware power. We were also trying to train a model based on big news articles with around 600-800 words with summaries with 80-120 words. The dataset was about 1.5 GBs which required a significant amount of memory. In future experiments we want to design a model using bi-directional RNN that uses SEQ-2-SEQ modeling logic to achieve better results on datasets with higher count of words

Conclusion: In this project we successfully predicted summaries based on the reviews that were given as an input to the model. We predicted the summaries with about 88% accuracy. We also inferred that a large number of epochs or batch size doesn't necessarily mean higher accuracy as the model might start overfitting with validation loss increasing over time.
Overall Text summarization is a great tool that can be used to summarize huge articles with a higher number of words as proved by the above project. Our model can be tweaked using Bi-directional LSTM that can produce good results for such use cases in future.

Reference:

1. https://aclanthology.org/D15-1044.pdf
2. https://aclanthology.org/P18-2027.pdf
3. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2749095.pdf
4. http://web.cs.wpi.edu/~claypool/mqp/sv/2018/juniper/juniper-final.pdf
5. https://www.ijcaonline.org/archives/volume176/number33/keerthana-2020-ijca-920401.pdf
6. Nallapati, Zhou, Santos, Gulçehre, & Xiang (2016)
7. Sutskever, I., Vinyals, O., & Le, Q. V. (2014)
8. Bahdanau, Cho, & Bengio, (2014)