Text Summarization Using Deep Learning

Parth Valani
Computer Science
Lakehead University 1116576
Thunder Bay, Canada
pvalani@lakeheadu.ca

Birva Patel
Computer Science
Lakehead University 1111092
Thunder Bay, Canada
bpatel24@lakeheadu.ca

Jaykumar Nariya
Computer Science
Lakehead University 1116571
Thunder Bay, Canada
jnariya@lakeheadu.ca

***Abstract*—Text summarization is considered as a challenging task in the NLP community. The availability of datasets for the task of multilingual text summarization is rare, and such datasets are difficult to construct. This article will show you how to work on the problem of text summarization to create relevant summaries for product reviews about fine food sold on the world's largest e-commerce platform, Amazon using basic character-level sequence-to-sequence (seq2seq) model by defining the encoder-decoder recurrent neural network (RNN) architecture. We implemented abstractive summarization in our paper. Abstractive text summarization aims to shorten long text documents into a human readable form that contains the most important facts from the original document.**

***Index Terms*—Text Summarization, Natural language processing, Abstrative, Extractive,LSTM,encoder-decoder.**

I. Introduction

Text summarization is a method in natural language processing (NLP) for generating a short and precise summary of a reference document. Producing a summary of a large document manually is a very difficult task. Summarization of a text using machine learning techniques is still an active research topic. Before proceeding to discuss text summarization and how we do it, here is a definition of summary.

A summary is a text output that is generated from one or more texts that conveys relevant information from the original text in a shorter form. The goal of automatic text summarization is to transform the source text into a shorter version using semantics.

There are approaches which are helpful to generate a summary – extraction and abstraction.As shown in Fig-1, Extraction is domain independent and identify the important sentences or phrases from the original text and extract only those from the text.

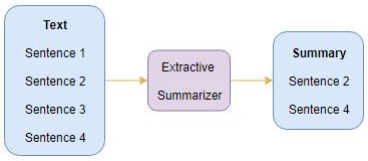


Fig. 1. Extractive Summarization.

while on the other hand, abstraction is domain dependent and takes the human knowledge by understanding the whole text and prepares a goal and produce a summary as shown in Fig-2.The sentences generated through abstractive summarization might not be present in the original text.

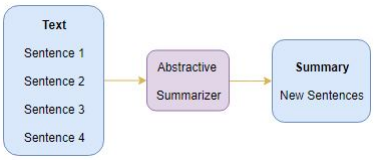


Fig. 2. Abstractive Summarization.

The paper organization is as follows. Section 2 describes the related work done by the pioneers in summarization. Section 3 provides the method used to implement text summarization. Section 4 includes the proposed model design. Section 5 indicates how we further improve the proposed model. section 6 shows the appendix work and section 7 is conclusion part.

II. Early Work

The earliest work on the summarization of a single document concentrated on technical documents. Perhaps the most cited paper on description is the one from (Luhn, 1958), which outlines research done at IBM during the 1950s [1]. Throughout his thesis, Luhn proposed that the frequency of a given word in an article would provide a useful measure of its significance.There are several primary ideas put forward in this paper that have taken on the value of summarization in later work. Words were truncated into their root forms as a first step, and stop words were deleted. Luhn then compiled a list of words of content sorted by decreasing frequency, the index giving a meaning measure of the word.On a sentence level, a significance factor was derived that reflects the number of occurrences of significant words within a sentence, and the linear distance between them due to the intervention of non-significant words. All sentences are ranked in order of their significance factor, and the top ranking sentences are finally selected to form the auto-abstract.

Baxendale in 1958 focused on sentence position to find the salient features [2]. He took 200 paragraphs and examined that in 85% of paragraphs topic sentences are placed in the beginning while in rest 7% he found, it occurred in the last.

Edmundson(1969) describes a system which produces excerpts from documents. His primary contribution was to develop a typical structure for an experiment with extractive

summarisation. [3]. At first, the author developed a protocol for creating manual extracts, that was applied in a set of 400 technical documents. From the previous two books, the two features of word frequency and positional value were integrated [3]. Two other features were used: the presence of cue words (presence of words like significant, or hardly), and the skeleton of the document (whether the sentence is a title or heading). Each of these features had weights added manually to score growing sentence. During evaluation, it was found that about 44% of the auto-extracts matched the manual extracts [4].

Research in the field of text summarization began in the 1950s and until now there is no system that can produce summaries such as professionals or humans.

III. Methodology

we developed a modeling pipeline and encoder-decoder architecture that tries to create relevant summaries for a given set of reviews. The modeling pipelines use RNN models written using the Keras functional API. The pipelines also use various data manipulation libraries.

The encoder-decoder architecture is used as a way of building RNNs for sequence predictions.This is because they are capable of capturing long term dependencies by overcoming the problem of vanishing gradient. It involves two major components:

- Encoder
- Decoder

The encoder reads the complete input sequence and encodes it into an internal representation, usually a fixed-length vector, described as the context vector. The decoder, on the other hand, reads the encoded input sequence from the encoder and generates the output sequence. Various types of encoders can be usedmore commonly, bidirectional RNNs, such as LSTMs, are used. We can build a Seq2Seq model on any problem which involves sequential information. This includes Sentiment classification, Neural Machine Translation, and Named Entity Recognition – some very common applications of sequential information.

Our objective is to build a text summarizer where the input is a long sequence of words (in a text body), and the output is a short summary (which is a sequence as well). So, we can model this as a Many-to-Many Seq2Seq problem. Below in Fig-3 is a typical Seq2Seq model architecture:

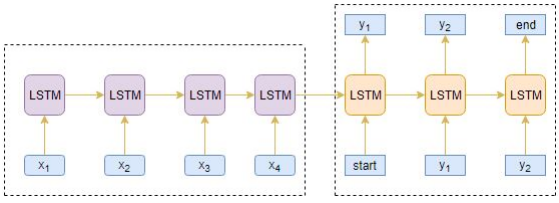


Fig. 3. Encoder-Decoder.

We can set up the Encoder-Decoder in 2 phases:

1) Training phase:In the training phase, we will first set up the encoder and decoder. We will then train the model to predict the target sequence offset by one timestep. Let us see in detail on how to set up the encoder and decoder.
   - encoder: An Encoder Long Short Term Memory model (LSTM) reads the entire input sequence wherein, at each timestep, one word is fed into the encoder. It then processes the information at every timestep and captures the contextual information present in the input sequence.
   - decoder: The decoder is also an LSTM network which reads the entire target sequence word-by-word and predicts the same sequence offset by one timestep. The decoder is trained to predict the next word in the sequence given the previous word.
2) Inference phase: As shown in Fig-4, After training, the model is tested on new source sequences for which the target sequence is unknown. So, we need to set up the inference architecture to decode a test sequence:

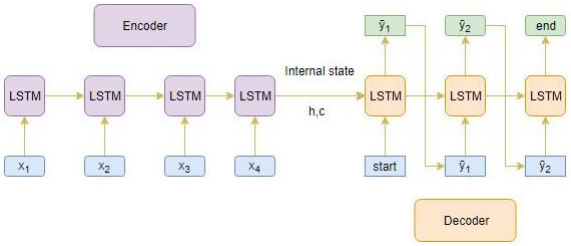


Fig. 4. Inference Process.

Here are the steps to decode the test sequence:

- Encode the entire input sequence and initialize the decoder with internal states of the encoder
- Pass $\langle start \rangle$ token as an input to the decoder
- Run the decoder for one timestep with the internal states
- The output will be the probability for the next word. The word with the maximum probability will be selected
- Pass the sampled word as an input to the decoder in the next timestep and update the internal states with the current time step
- Repeat steps 3 – 5 until we generate $\langle end \rangle$token or hit the maximum length of the target sequence

IV. Proposed Model

Here, In Fig-5 we explained the our flow of the model.

- Custom Attention Layer: Keras does not officially support attention layer. So, we can either implement our own attention layer or use a third-party implementation. We will go with the latter option for this article.copy it in a different file called attention.py and import it into our environment

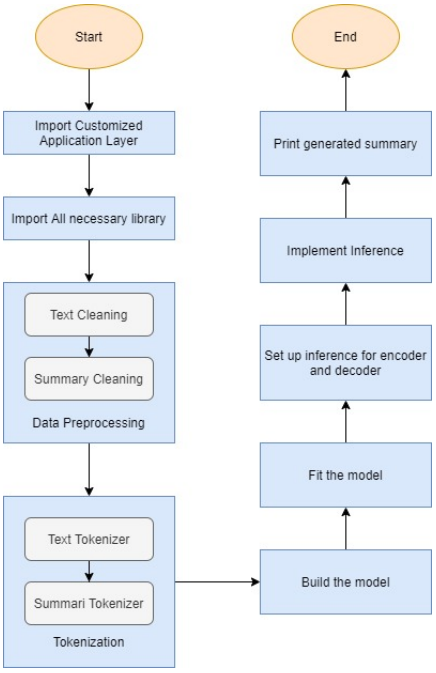


Fig. 5. Flow of the Proposed Model.

- Import the Libraries: we can call numpy for array operations, pandas for data visualization, BeautifulSoup for parsing HTML and XML documents and some keras models and layers.
- Read the dataset: This dataset consists of reviews of fine foods from Amazon. The data spans a period of more than 10 years, including all 500,000 reviews up to October 2012. These reviews include product and user information, ratings, plain text review, and summary. It also includes reviews from all other Amazon categories.We'll take a sample of 100,000 reviews to reduce the training time of our model.
- Drop Duplicates and NA values: drop the duplicate text that causes a data-length problem.
- dictionary to expand the contractions: here we showed only some part of the proposed dictionary.
- Text Cleaning: We will perform the below preprocessing tasks for our data:
  - Convert everything to lowercase
  - Remove HTML tags
  - Contraction mapping
  - Remove ('s)
  - Remove any text inside the parenthesis ( )
  - Eliminate punctuations and special characters
  - Remove stopwords
  - Remove short words
- Summary Cleaning:now we'll look at the first 10 rows of the reviews to an idea of the preprocessing steps for the summary column.
- Drop empty rows:
- distribution of the sequences: we will analyze the length of the reviews and the summary to get an overall idea about the distribution of length of the text As shown in Fig-6. This will help us fix the maximum length of the sequence.

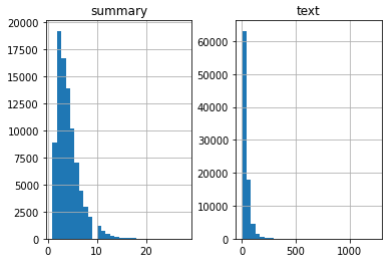


Fig. 6. Distribution of length.

- Fix the Maximum Length: We observe that 94% of the summaries have length below 8. So, we can fix maximum length of summary to 8.Let us fix the maximum length of review to 30
- Add the start and end tokens:
- Split the dataset: we need to split our dataset into a training and validation set. We'll use 90% of the dataset as the training data and evaluate the performance on the remaining 10%.
- text tokenizer: A tokenizer builds the vocabulary and converts a word sequence to an integer sequence.same for the summary tokenizer as shown in appendix-
- Row deletion:deleting the rows that contain only START and END tokens
- model building: build a 3 stacked LSTM for the encoder
- compile and fit: .I am using sparse categorical cross-entropy as the loss function since it converts the integer sequence to a one-hot vector on the fly. This overcomes any memory issues. And Earlystopping is used to stop training the neural network at the right time by monitoring a user-specified metric. Here, I am monitoring the validation loss (val_loss). Our model will stop training once the validation loss increases.We'll train the model on a batch size of 128 and validate it on the holdout set (which is 10% of our dataset)
- Accuracy graph: As shown in Fig-7, we will plot a few diagnostic plots to understand the behavior of the model over time.We can infer that there is a slight increase in the validation loss after epoch 10. So, we will stop training the model after this epoch.
- convert index to word: build the dictionary to convert the index to word for target and source vocabulary
- Encoder and decoder for Inference:
- implement inference process:
- Convert to Word sequence: Define the functions to convert an integer sequence to a word sequence for summary as well as the reviews.
- Print Summary: Print first 100 reviews to check summaries generated by the model.

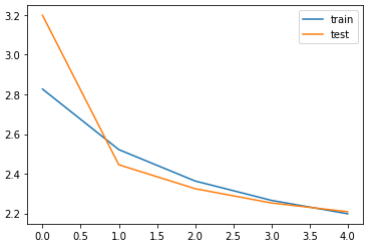


Fig. 7. Accuracy Comparison.

V. Improvement in Model Performance

There's a lot more we can do to play around and experiment with the model:

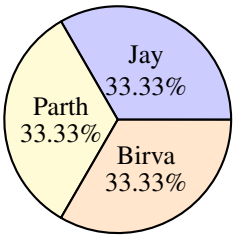
- I recommend you to increase the training dataset size and build the model. The generalization capability of a deep learning model enhances with an increase in the training dataset size
- Try implementing Bi-Directional LSTM which is capable of capturing the context from both the directions and results in a better context vector
- Use the beam search strategy for decoding the test sequence instead of using the greedy approach (argmax)
- Evaluate the performance of your model based on the BLEU score
- Implement pointer-generator networks and coverage mechanisms

VI. Conclusion

In our model, We used different types of deep convolution networks. First, We used simple convolution network(containing Conv1D,max-pooling,relu and dense layers) but we got very low accuracy around 30-40% on the same epoch as our model. So here We implemented it using a sequence-to-sequence (seq2seq) model by defining the encoder-decoder(with attention) recurrent neural network(RNN) architecture.as a result we are getting the summary of the amazon fine food ratings that includes product and user information, ratings, and a plain text review and we got 64.43% accuracy at 17 epochs. There are many other technique also available like extractive summrization. but this gives us more accurate result.

VII. Contribution

As expected, we as a team performed very well for this project Also we distributed the work in same proportion and most importantly, completed it on time without any complications.



References

[1] H. P. Luhn, "The automatic creation of literature abstracts," *IBM Journal of research and development*, vol. 2, no. 2, pp. 159–165, 1958.

[2] P. B. Baxendale, "Machine-made index for technical literature—an experiment," *IBM Journal of research and development*, vol. 2, no. 4, pp. 354–361, 1958.

[3] H. P. Edmundson, "New methods in automatic extracting," *Journal of the ACM (JACM)*, vol. 16, no. 2, pp. 264–285, 1969.

[4] O. Tas and F. Kiyani, "A survey automatic text summarization," *PressAcademia Procedia*, vol. 5, no. 1, pp. 205–213, 2007.