Abstractive text summarization using recursive RNN model

News headline generator

Arpan Ghoshal
Computer Science Engineering
PES University
Bangalore, India
arpanghoshal@outlook.com

Nikunj Goyal
Computer Science Engineering
PES University
Bangalore, India
nikunjgoyal58@gmail.com

Abstract—**There are many techniques of text summarization to give an output which will give the required text needed for the understanding of an article. We have implemented a model that involves technologies like recursive recurrent neural networks and long short term memory. These are the deep learning techniques that use attention-based mechanisms to compute. It can be easily trained by a large amount of data as an input. We have used news article data set from CNN, which showed better performance on the models implemented. In this paper, we also examine how the model affects the output generated.**

Keywords— Text summarisation, Deep learning, RNN, LSTM, encoder-decoder, seq-2-seq, News headline generator

I. INTRODUCTION

Summaries reduce reading time. Text summarization is the tool for easing the task of reading long articles which are time-consuming. Summarizing a text can happen in an extractive way where only sentences involving the keywords are prioritized and is shown as the output. But this will not the user-friendly sentence required for news headlines to be. An approach is needed where the output should look like it is a human written format. This approach can be implemented by deep learning techniques like neural networks which are the set of algorithms inspired by the human brain. These algorithms are used to recognize the patterns in the particle.

Recurrent neural networks, improvised version of neural network which is used for the sequential data to process. These are the only algorithms that have internal memory for the process. Because of this RNN is better in remembering important things in the text. This mechanism can be used in various applications like speech, audio, video, weather, data, and the most needed text. It enables for precise prediction of what coming next in the sequence.

Long short term memory is an improvement over RNN which prioritizes the embedding for predicting. To add a piece of new information in the memory it undergoes the existing information to the function compiled by transiting functions. We will implement the model by libraries like Keras, which is a high-level API for neural networks and TensorFlow as the dataset used is very large.

The recursive model which is used by us is the most efficient in the term of output and accuracy shown. It recursively implements the RNN model with LSTM hidden layer. This model is further discussed elaborately as it is the best model of RNN to be implemented for this type of input-output model.

These models help to fulfill our aim to produce an output which is like a human-generated summary. The summary should be short and concise as it is the news headline of an article.

The remainder of the paper is structured as follows: Section II shows the types of implementation of text summarization done and why it has not proceeded further. Section III pictured the models of summarizer which can be used. Section IV shows the evaluation of the method possessed and the results obtained. Section V ends with the conclusion and scope for further work.

II. SUMMARIZATION TECHNIQUES USED

We have implemented two types of text summarization. One gave the keywords for making the output sentence which gave important words and others gave the output text like a human-written format.

Two types which we implemented are---

A. Extractive Summarization

Textrank algorithm was used to extract the most important sentences in the article. It is a graph-based, surface-level algorithm. In this, the similarity of weights of vertices in edges is used. The score is measured by the links made in it. Stopwords removed by NLP techniques by using libraries such as NLTK.

Consider a graph G = (V, E), V is a set of vertices and E is a set of edges.

$$S(V_i) = (1-d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

Where E is a subset of V x V, for V_i . ln(V_i) is representing a subset of predecessors of vertices and Out(V_i) is a set of successors.

$$Similarity(S_i, S_j) = \frac{|\{w_k | w_k \in S_i \& w_k \in S_j\}|}{log(|S_i|) + log(|S_j|)}$$

The similarity of the words is found by TF-IDF(.Term Frequency-Inverse Document Frequency), cosine similarity, etc. Once the ranking algorithm is run with the input graph, the sentences are sorted in descending order of their score and the highest ranked sentences are selected to build the summary.

First, we represented the document as numerical vectors, where each index is corresponding to the index of vocabulary set. The formula for TF-IDF is:

$$tf(w, d) = f_d(w): \text{ frequency of } w \text{ in document } d$$
$$idf(w, D) = log \frac{1+|D|}{1+df(d,w)}$$

We simply put TF-IDF values instead of just counting by n-grams, appear in a sentence.

B. *Abstractive Summarization*

Textrank algorithm was implemented properly but it just extracted phrases needed from an article. So the abstractive model is used with the help of the neural network and hidden long short term memory. ROUGE score for abstractive ones is much higher than extractive.

It requires training of data, a large number is preferable. During training, it will optimize the efficiency of summarised text. Both the encoding layer and language model are trained simultaneously.

We have further implemented this by the models discussed in section III.

III. MODELS IMPLEMENTED

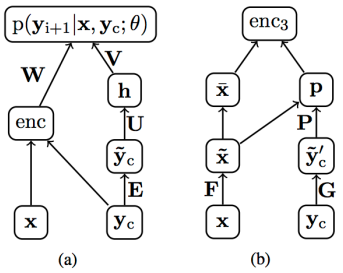
The encoder-decoder recurrent neural network implemented for machine translation proved to be better for text summarization.

A. *Encoder-Decoder Architecture*

This is a way for sequence prediction problems that have several inputs, outputs and inputs, and outputs. Encoders help in reading the entire text of input and encodes it into a representation, a fixed length vector known as context vector. Decoder reads the output of the sequence generated by encoders.

Encoders can be used in different ways, most commonly bidirectional RNN's such as LSTM's, are used. Word embedding like GloVe is used in encoders for the representation of words.

The context vector can be fixed encoding as in the simple Encoder-Decoder architecture or it can be more expressive by attention mechanism.



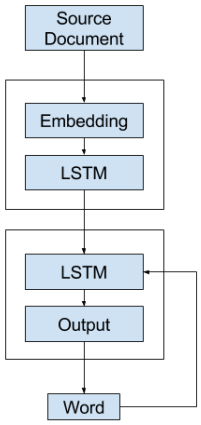
In this representation, x is the source document, enc is encoder which is implying an internal representation of x, and yc is the sequence of previously generated words.

The decoder takes hidden layers generated after feeding the last word of input text as input. First, the symbol of end sequence is fed as input, again embedded layer is used to transfer the symbol to a distributed representation.

B. *Implementation model*

Now, we implement Encoder-Decoder architecture for text summarization using Keras and tensorflow deep learning libraries. The models are described below.

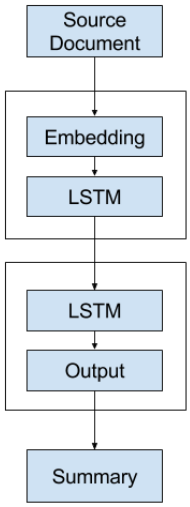
1) General model: The implementation of the model involves an Encoder with LSTM hidden layer that produces a fixed-length implementation of the input document. The Decoder reads the output generated by the embedding of each last generated word in the output summary.



While implementing this a problem occurred. Keras does not allow any recursive loops where the input of the model is taken from output automatically. So we took a different approach which can be implemented in Keras.

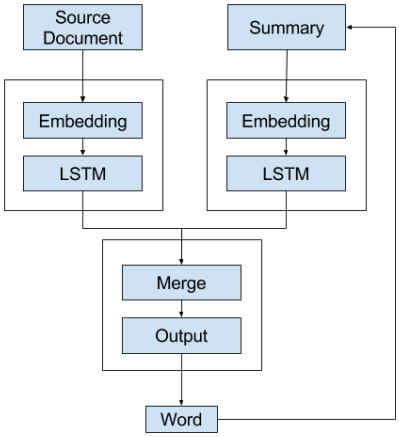
We have integrated Keras with tensorflow for computing better. Seq 2seq model is used in our case.

2) One-shot model: This is the alternative model we tried to implement where the output sequence is generated in a one-shot manner. The output sequence is generated by context vectors with the help of decoders.



It is possible that the decoder is not provided with sufficient input context for output sequence to be generated as it has chosen the word and its order.

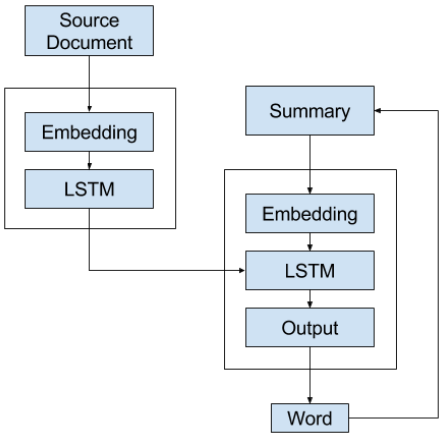
3) Recursive model I*:* This is a second alternative model where we are generating a single word forecast and calling it recursively. The decoder uses the whole output by context vectors and distributed representations as input for predicting the next word. A model of language can be used for interpreting the sequence of words till now to provide a next context vector for a combination of representation of source document to generate the next word.



A summary is recursively built by calling the model with words generated previously (words trained during the training of the model). The context vectors are added or concatenated to make a more spectrum of context providing to the decoder predict next words.

This is better as decoder have a chance to use words generated before and the input document for predicting next words. But it puts the burden on the merge function and decoder itself to where the output sequence is heading to.

4) **Recursive model II***:* This is the best model presently which we have implemented for making RNN and LSTM into reality. In each phase of the generated output sequence, the document is provided to the decoder. This helps in allowing the decoder for building same internals states which were used for generating the word in the output sequence.



This process is repeated many times by calling the model recursively for words in output sequence until the end of sequence or maximum of length is generated.
This concludes our exploration of models as this provided the best outcome with very fewer limitations.

IV. EVALUATION AND RESULTS OBTAINED

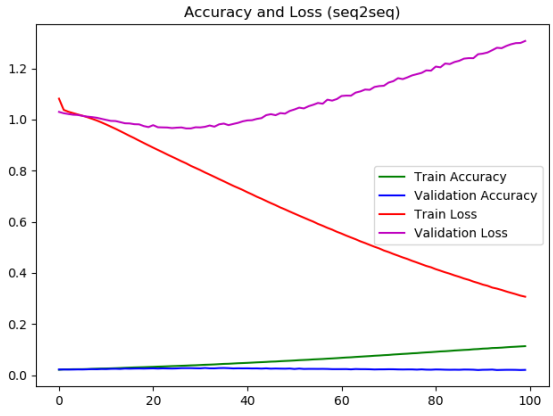
Evaluation of the models is done on CNN dataset is taken from kaggle where there are headlines associated with the articles. The input sequence is then transferred through the model in training dataset in a different number of epochs depending on the model. While implementing we have used one hot encoding for the structure.

While training a machine learning model, overfitting is the main thing which should be avoided. When the model fits with training data well but it is not efficient in generalizing and making a prediction of data which is not given as input before.

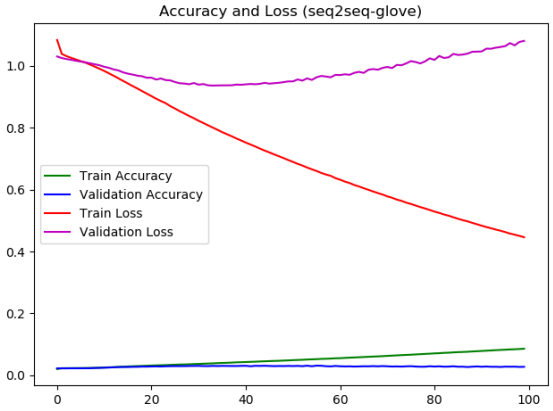
To find out the model is overfitting are not, a technique is used called cross-validation where the data is split into a training set and validation set. Training sets are used to train

the model while validation for evaluating the model's performance which can predict inputs which have not occurred before.

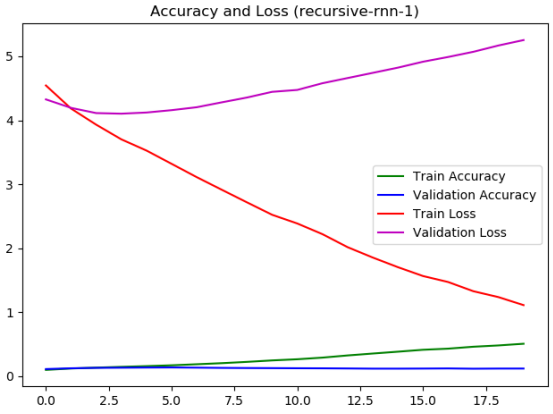
Loss and acc are measures of loss and accuracy on the training set. val_loss and val_acc are measures of loss and accuracy of new data.



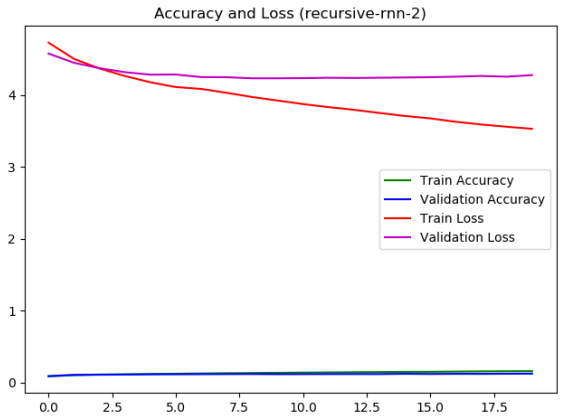
The above graph is of the one-shot model implementation of RNN. It encodes the whole content of the article and decodes it in one go. As we can observe from the graph created by matplotlib, that training accuracy is very less so we tried glove embedding to notice any change and improve efficiency.



But there was no improvement in accuracy. But a slight improvement in the loss. So we needed a special type of model to increase the accuracy. We implemented merge type of recursive recurrent neural network which showed load in merge operation where it takes current built up text to predict next.



Recursive RNN in which it takes the article content and the current built up text to predict the next character of text with one layer of LSTM decoder, is used and every limitation possessed earlier showed significant improvement.



This is the final result obtained with an accuracy of 43.5%. With a slight fraction of errors in output which made this project as a developing where it can be improved more further.

We have used TF-IDF for focusing on important words. Important but rare words by switching the decoder and pointer. Keras and tenser flow are the libraries which helped in the computation of deep learning techniques. As large dataset is used tenser flow significantly helped the process. Other data manipulation libraries like scikit-learn, pandas and lumpy were also used. These python libraries helped in improving the efficiency of summarization of an article.

V. CONCLUSION AND FUTURE WORK

In this paper, the models are described for summarising an article, in our case, it is summarising article into one sentence making as a headline of the article. First, it is done by extractive type but it was showing only the important phrases from the article which gave repeated words or unformatted lines. So we tried abstractive form. But in this also we have seen many complications and finally used a recursive recurrent neural network. This gave a better result and we proceeded further.

But the output generated can only be of the same category input, as the accuracy is not enough high which can give a somewhat accurate result on any input provided for text summarization. The headlines generated are not well structured and have grammatical errors. These problems can be minimized by converging some models.

In the first stage of the project, we tried to implement a model where the output of extractive text summarization can be directed to the RNN + LSTM model. This can give the abstractive summary as the concise one with important words and human-readable format.

REFERENCES

[1] M. Allahyari et al., "Text Summarization Techniques: A Brief Survey", *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 10, 2017. Available: 10.14569/ijacsa.2017.081052.

[2] S. Song, H. Huang and T. Ruan, "Abstractive text summarization using LSTM-CNN based deep learning", *Multimedia Tools and Applications*, vol. 78, no. 1, pp. 857-875, 2018. Available: 10.1007/s11042-018-5749-3.

[3]*Ijarcsse.com*, 2019. [Online]. Available: http://ijarcsse.com/Before_August_2017/docs/papers/Special_Issue/iconect2016/sfcs02.pdf. [Accessed: 17- May- 2019].

[4]K. Knight and D. Marcu, "Summarization beyond sentence extraction: A probabilistic approach to sentence compression", *Artificial Intelligence*, vol. 139, no. 1, pp. 91-107, 2002. Available: 10.1016/s0004-3702(02)00222-9.

[5]X. Zuo, S. Zhang and J. Xia, "The enhancement of TextRank algorithm by using word2vec and its application on topic extraction", *Journal of Physics: Conference Series*, vol. 887, p. 012028, 2017. Available: 10.1088/1742-6596/887/1/012028.

[6]Lai, Siwei, Liheng Xu, Kang Liu and Jian Zhao. “Recurrent Convolutional Neural Networks for Text Classification.” AAAI (2015).

[7] Douma, Nejmeddine. “Attention Neural Network-Based Abstractive Summarization and Headline Generation.” (2018).