RNN based article caption generator

Project report for class CSCE-689 (Natural Language Processing), Spring 2018

Instructor - Professor Ruihong Huang

Abhinav Ratna
Electrical & Computer Engineering
Texas A&M University
aratna@tamu.edu
UIN-227001723

Palash Parmar
Electrical & Computer Engineering
Texas A&M University
palparmar@tamu.edu
UIN-626008848

***Abstract*— We proposed a title generation method with a recurrent neural network using concepts of machine translation. The title generator consists of an encoder and a decoder and they are constructed with Long Short Term Memory, which is one of the recurrent neural networks. We construct a distributed representation of an article in the encoder and the decoder generates a title without extraction of an article. In some evaluational experiments we confirmed that our proposed method could generate appropriate titles from articles but in few articles the method generate random titles.**

***Index Terms*— Natural Language Processing, LSTM, Attention, RNN, Recurrent Neural Network, Word Embedding.**

I. INTRODUCTION AND MOTIVATION

With the recent increase in the amount of content available online,fast and effective automatic summarization has become more important. The need for getting maximum information by spending minimum time has led to more efforts being directed to the field of summarization. Most current automatic summarization systems use sentence extraction, where key sentences in the input documents are selected to form the summary. The key challenge in summarization is to optimally compress the original document in a lossy manner such that the key concepts in the original document are preserved. Off-the-shelf attentional encoder-decoder RNN that was originally developed for machine translation have recently been found to be very effective for many transduction tasks- that is transforming text from one form to another. In this project, we aim to implement encoder-decoder recurrent neural network with LSTM units to generate title for short text articles.

II. PROBLEM ANALYSIS

The objective of the article summarization task is to generate a condensed summary from an input sentence. A article is a sequence of N words and the summary generated is a sequence of M words where $M < N$.

III. RELATED WORKS

There have been many approaches developed over the past half century to generate good quality summaries from the text. Researchers started with linguistic features of the English text to find out which sentences are more important and included them in the summary. But it was hard for researchers to write a code that works with different types of writing styles of the authors. To make a generic summarization system algorithms were developed based on statistical properties of the language such as word frequencies. Until now the focus has been to extract content from the text as a summary. Extractive text summarization techniques rely on text features such as position in the text, word frequency, information content of the sentence to decide whether to add that portion of text in summary or not. Most commercially available summarization systems utilize the extraction based approach. They crop out important phrases from the text and glue them together to generate the meaningful summary. Although the

summary generates grammatical sentences, it fails to capture the meaning in fewer words.

Recently, Recurrent neural networks have recently been found to be very effective for many transduction tasks - that is transforming text from one form to another. Examples of such applications include machine translation and speech recognition. These models are trained on large amounts of input and expected output sequences, and are then able to generate output sequences given inputs never before presented to the model during training.

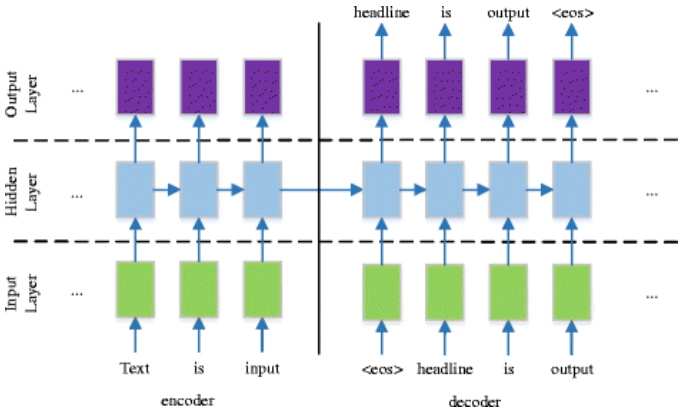

Fig. 1. Encoder decoder architecture

IV. METHODOLOGY

A. Dataset

1) Source: The model is trained on the "Signal Media One-Million News Articles Dataset". This dataset contains 1 million articles that are mainly English , but they also include non-English and multi-lingual articles. Each of the news articles has a clearly delineated headline and text, where the text is broken up into paragraphs.

2) Preprocessing: The headline and text are lowercased and tokenized, separating punctuation from words. Only the first paragraph of the text is kept. An end-of-sequence token is added to both the headline and the text. Articles that have no headline or text, or where the headline or text lengths exceed 25 and 50 tokens, respectively, are filtered out, for computational efficiency purposes. All rare words are replaced with the <unk> symbol, keeping only the 40,000 most frequently occurring words. We split the dataset into training and test set keeping 1000 examples in test set.

B. Model Architecture

1) Overview: The architecture is comprised of two parts encoder and decoder recurrent neural networks as shown in below figure.

The encoder is fed as inputting the text of a news article one word of one time. Each word first passes through an embedding layer that transforms the word into a distributed representation. This distributed representation is then combined using a multi-layer neural network with the hidden layers generated after feeding in the previous word, or all 0s for the first word in the text.The decoder takes as input the hidden layers generated after feeding in the last word of the input text. First, an end-of-sequence symbol is fed in as input, again using an embedding layer to transform the symbol into a distributed representation. Then, the decoder generates the text summaries, using a softmax layer and the attention mechanism described in the next section, each of the words of the headline, ending with an end-of-sequence symbol. After generating each word, the same word is fed in as input when generating the next word. The loss function we use is the log loss function:

$$
\begin{aligned}
&-\log p\left(y_1, \ldots, y_T | x_1, \ldots, x_T\right) = \\
&-\sum_{t=1}^{T} \log p\left(y_t | y_1, \ldots, y_{t-1}, x_1, \ldots, x_T\right)
\end{aligned} \tag{1}
$$

During testing we use a beam-search decoder which generates input words one at a time, at each step extending the B highest probability sequences.We use 4 hidden layers of LSTM units, Each layer has 512 hidden units. We have also used dropout for regularization to reduce overfitting so that training and testing performance could be similar.

We have used a learning rate of 0.01 along with the RMSProp and adam both gradient optimization methods. . For RMSProp we use a decay of 0.9 and a momentum of 0.9. We train for 9 epochs, starting to half the learning rate at the end of each epoch after 5 epochs. We processed 468 batch examples at a time. This batching complicates the implementation due to the varying lengths of different sequences. We simply fix the maximum

lengths of input and output sequences and use special logic to ensure that the correct hidden states are fed in during the first step of the decoder, and that no loss is incurred past the end of the output sequence. Overall model summary can be understood from below diagram (figure 2).

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 50, 100)	4000000
lstm_1 (LSTM)	(None, 50, 512)	1255424
dropout_1 (Dropout)	(None, 50, 512)	0
lstm_2 (LSTM)	(None, 50, 512)	2099200
dropout_2 (Dropout)	(None, 50, 512)	0
lstm_3 (LSTM)	(None, 50, 512)	2099200
dropout_3 (Dropout)	(None, 50, 512)	0
simplecontext_1 (Lambda)	(None, 25, 944)	0
time_distributed_1 (TimeDist	(None, 25, 40000)	37800000
activation_1 (Activation)	(None, 25, 40000)	0

Total params: 47,253,824
Trainable params: 47,253,824
Non-trainable params: 0

Fig. 2. Attention mechanism architecture

2) Attention: Attention is a mechanism that helps the network remember certain aspects of the input better, including names and numbers. The attention mechanism is used when outputting each word in the decoder. For each output word the attention mechanism computes a weight over each of the input words that determines how much attention should be paid to that input word. The weights sum up to 1, and are used to compute a weighted average of the last hidden layers generated after processing each of the input words. This weighted average, referred to as the context, is then input into the softmax layer along with the last hidden layer from the current step of the decoding.

Similar to the basic encoder-decoder architecture, the attention mechanism plug a context vector into the gap between encoder and decoder. According to the schematic shown in figure 3, blue represents encoder and red represents decoder; and we could see that context vector takes all cells outputs as input to compute the probability distribution of source language words for each single word decoder wants to generate. By utilizing this mechanism, it is possible for decoder to capture somewhat global information rather than solely to infer based on one hidden state. Building context

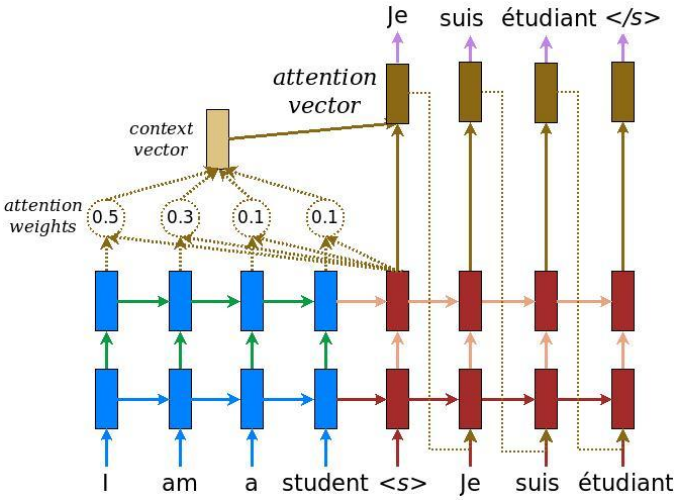

Fig. 3. Attention mechanism architecture

vector is also fairly simple. For a fixed target word, first, we loop over all encoders states to compare target and source states to generate scores for each state in encoders. Then we could use softmax to normalize all scores, which generates the probability distribution conditioned on target states. At last, the weights are introduced to make context vector easy to train. The attention weight for the input word at position t computed when outputting the t'-th word is:

$$a_{y_{t'}}(t) = exp(h_{x_t}^T h_{y_{t'}}) / \sum_{\bar{t}}^{T} exp(h_{x_{\bar{t}}}^T h_{y_{t'}}) \qquad (2)$$

where h_{x_t} represents the last hidden layer generated after processing the t-th input word, and $h_{y_{t'}}$ represents the last hidden layer from the current step of decoding. Note one of the characteristics of this mechanism is that the same hidden units are used for computing the attention weight as for computing the context.

V. RESULTS

A. Evaluation

The model was evaluated using BLEU (bilingual evaluation understudy) evaluation metrics. BLEU is an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another. The BLEU evaluation metric looks at what fraction of n-grams of different lengths from the expected headlines are actually

output by the model. It also considers the number of words generated in comparison to the number of words used in the expected headlines. Average BLEU score for 400 test examples over 10 epochs is 0.06. Training loss with epochs can be seen from below diagram.

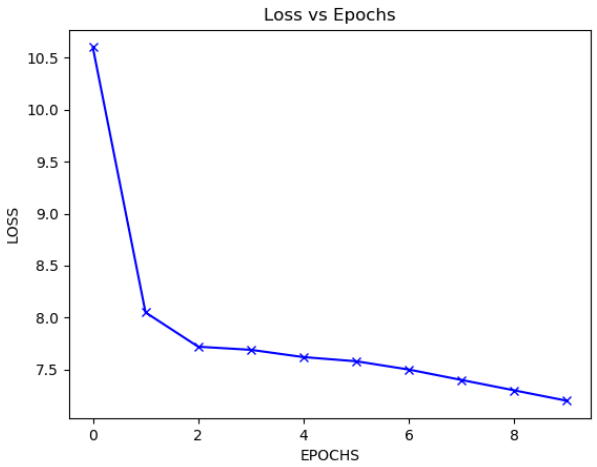

Fig. 4. Training loss with epochs

B. Analysis

While predicting the headlines, we observed that the model is quite effective in predicting the headlines of the articles it was trained on. For test example, model shows mixed performance. For few of the text examples, summary generated is quite close to the actual headline while for few it is way off. Model took roughly 3 days to train on Nvidia Tesla K80 GPU for 200 epochs. Few of the examples can be seen from following table.

Article	Actual Headline	Generated Headline
'We're too broke to go on the beat': Police chiefs warn ministers that cuts mean they can't do their job as they threaten to stop street patrols and say they won't be able to protect public from rioters or terrorists?	Police chiefs warn ministers Were too broke to go on the beat	Police chiefs say not able to protect public warn ministers
What have you been listening to this year ? If you want to find out using cold , hard evidence , then Spotify 's new Year in Music tool will tell you .	Spotify Will Make You Smarter for Your App	The 10 Most Popular Songs Of All Time
NHS patients to be given option of travelling to Calais for surgical procedures NHS patients in Kent are set to be offered the choice of travelling to Calais for surgical treatments, local health commissioners have confirmed.	NHS patients to be given option of travelling to Calais for surgical procedures	Patients travelling Calais

Fig. 5. Actual vs Predicted headline

VI. BOTTLENECKS

A. Dataset processing

Many examples in ”Signal Media One-Million News Articles Dataset” contain very big articles and their headline does not summarize the text well. Apart from this, many articles have headlines in coded form with many special symbols in it.

B. Model Training

Since the dataset contains 1 million articles so training the model with LSTM units RNN model and attention mechanism takes huge time. Our model took around 3 days (with 200 epochs) to train on a Nvidia Tesla K80 GPU.

VII. CONCLUSION AND FUTURE WORK

We have successfully implemented our encoder decoder recurrent neural network with LSTM units and attention model for generating headlines from the articles of ”Signal Media One-Million News Articles Dataset”. The model performs better on the articles from the dataset than any general test article. Most of the times, model generates valid and grammatically correct summary. we have also been able to implement attention mechanism in the model which helps model in deciding which words of the input text to pay attention to when generating each output word. With this mechanism, We find that the network learns to detect linguistic phenomena, such as verbs, prepositions, negations etc.

In future, we aim to improve the model by using certain techniques such as bidirectional RNN which is seen to be working better with attention mechanism. We are also planning to get rid of bottlenecks described in previous section by using smaller dataset and state-of-the-art deep learning techniques.

ACKNOWLEDGMENT

We would like to thank Professor Ruihong Huang, Department of Computer Science and Engineering (Texas A & M University) for her support and encouragement throughout the project. Her valuable feedback helped a lot in execution of this project.Finally, we would like to say thanks to Konstantin Lopyrev, the author of ”Generating News Headlines with Recurrent Neural Networks”

paper. Many of the ideas implemented in our model is referred from his paper.

REFERENCES

[1] Ian Goodfellow, Aaron Courville, and Yoshua Bengio. Deep learning. Book in preparation for MIT Press,2015.

[2] Bing L, Li P, Liao Y et al (2015) Abstractive multi-document summarization via phrase selection and merging[J]. arXiv preprint arXiv:1506.01597

[3] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. CoRR, abs/1409.3215, 2014.

[4] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. CoRR , abs/1506.03099, 2015

[5] Cao Z, Li W, Li S et al (2016) Attsum: joint learning of focusing and summarization with neural attention[J]. arXiv preprint arXiv:1604.00125.

[6] Chen M, Weinberger KQ, Sha F (2013) An alternative text representation to TF-IDF and Bag-of-Words[J]. arXiv preprint arXiv:1301.6770.

[7] Andrej Karpathy and Fei-Fei Li. Deep visual-semantic alignments for generating image descriptions.CoRR, abs/1412.2306, 2014.

[8] Chopra S, Auli M, Rush AM (2016) Abstractive sentence summarization with attentive recurrent neural networks[C]. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp 9398.

[9] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5 - rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[10] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. CoRR , abs/1508.04025, 2015.