Two-Encoder Pointer-Generator Network for Summarizing Segments of Long Articles

Junhao LI† and Mizuho IWAIHARA‡

Graduate School of Information, Production and Systems
Waseda University, Kitakyushu, Japan
E-mail: † lijunhao@toki.waseda.jp, ‡ iwaihara@waseda.jp

Abstract Usually long documents contain many sections and segments. In Wikipedia, one article can usually be divided into sections and one section can be divide into segments. But although one article is already divided into smaller segments, one segment is often still too long to read. So, we consider that segments should have a short summary for readers to grasp a quick view of the segment. This paper discusses applying neural summarization models including seq2seq model and pointer generator network model to segment summarization. These models for summarization can take target articles as the only input to the model. However, in our case, it is very likely that other segments in the same article contain descriptions related to the target segment. Therefore, we propose several ways to add additional information from the whole article to the short target segment, as the input for summarization. We compare the results against the original models without additional information. Furthermore, we propose a new model that uses two encoders to process target segments and additional sequences separately. Our results show the two-encoder model outperforms the original models in terms of ROGUE and METEOR scores.

Keyword text summarization, deep learning, seq2seq, pointer generator network, multi-encoder

1. INTRODUCTION

Wikipedia, started from 2001, has been emerging as the world's largest encyclopedia. The English version of Wikipedia contains 5.7 million articles and over 600 articles are newly created in one day as of August 2018. Since the encyclopedia is managed by Wikipedia Foundation, an international non-profit organization, and a great number of collaborators in the world under some international projects, its articles are continuously edited and developed. Therefore, its content is quite reliable regardless of its openness [5].

With the number of Wikipedia articles growing very quickly, the information that Wikipedia contains is invaluable. But generally, the usage of Wikipedia is limited to mainly for human readers. The most important work that computer systems can support in using Wikipedia resource is searching and showing interested articles to readers [13]. With the continuously growing number of articles in Wikipedia, the search results might be very long. In additional, segments in articles sometimes become longer, which is making it difficult for human readers to grasp the whole idea from one specific segment. Summarizing the whole article may be one solution for this problem. But the summary of article contains information from the whole article, which would make it not suitable for the reader who only looks for information in several segments or sections. Also, the summary of one article may lose too much information from target segments or sections. So we think it is necessary to focus on segment summarization.

There are two broad approaches to summarization: Extractive and abstractive.

Extractive methods involve the selection of the phrases and sentences from the source document to make up new summary. Techniques involve ranking the relevance of sentences or phrases in order to choose only those most relevant to the meaning of the source. One widely used example is TextRank [17] that uses graph-based ranking algorithm to extract important sentences from a single document.

Abstractive methods attempt to develop an understanding of the main concepts in a document and then express those concepts in clear natural language. It uses linguistic methods to examine and interpret the text and then to find the new concepts and expressions to best describe it by generating a new shorter text that conveys the most important information from the original text document [19]. Recently deep learning methods have shown promising results for text summarization. The most popular model is sequence to sequence attention model [9].

In this paper, we mainly focus on summarization methods that involves deep learning models which contains both approaches. Since we need to add additional

information from the whole article and sometimes the whole article is too long containing too much noisy information. So we are using extractive methods to shorten the article into several sentences as additional information to the model. The rest of this paper is organized as follows. Section 2 covers related work. Section 3 is models we used. Section 4 shows three additional sequence methods. Section 5 is the results of our experiments and section 6 make discussions about it. Section 6 concludes this paper.

2. RELATED WORK

Long short-term memory (LSTM) was first proposed by Seep Hochreiter and Jürgen Schmidhuber [18] and improved in 2002 by Felix Gers' team [6]. LSTM networks are well-suited to classifying, processing and making predictions based on time series data. LSTMs were developed to deal with the exploding and vanishing gradient problems that can be encountered when training traditional RNNs. Later Graves et al. [7] proposed a neural network model using bidirectional LSTM (Bi-LSTM) for phoneme classification, showing great improvement over original LSTM.

Neural network-based Seq2Seq learning has achieved remarkable success in various NLP tasks, including but not limited to machine translation and text summarization. The Seq2Seq model is firstly proposed by Ilya et al. [2]. The Seq2Seq model is an encoder-decoder model such that the encoder converts the input sequence into one context vector that contains all information in the input sequence. Then the decoder generates the output sequence based on the context vector.

Bahdanau et al. [4] were the first to introduce attention mechanism to the seq2seq model to release the burden of compressing of entire source into a fixed-length vector as context. Instead, they proposed to use a dynamically changing context vector h_t^* in decoding process. e_i^t is the attention energy of hidden states of encoder at timestep i in decoder step t. a^t is the attention weights distribution. v^T, W_h, W_s and b_{atten} are learnable parameters.

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + b_{attn}) \quad (1)$$

$$a^t = \text{softmax}(e^t) \quad (2)$$

$$h_t^* = \sum_i a_i^t h_i \quad (3)$$

Rush, Chopra, et al. [1] were the first to apply Seq2Seq and attention model on text summarization, achieving state-of-the-art performance on DUC-2004 and Gigaword, two sentence-level summarization datasets.

Based on Rush [1], Luong et al. [14] proposed multi-task Seq2Seq model to solve different NLP problems using the same model in the same time. Their model settings include one encoder to many decoders, many encoders to one decoder and many encoders to many decoders. Their results show that it is a promising model for solving multi-task problems. Several results that use multiple encoders are even better than only one encoder.

Vinyals et al. [15] proposed a Seq2Seq model called pointer network which produces an output sequence consisting of elements from the input sequence applying soft attention distribution of Bahdanau et al. [4]. The pointer network has been utilized to create hybrid approaches for NMT, language modeling and summarization.

Abigail et al. [2] proposed a pointer-generator network model that combines pointer network with original Seq2Seq + attention model. Their model allows both copying words through pointing and generating words from a fixed vocabulary. In their model, they are using Bahdanau attention mechanism [4] in equations (1) (2) and (3) to calculate attention distribution a^t and context vector h_t^*. Their generation probability $p_{gen} \in [0,1]$ for timestep t is calculated from context vector, decoder state s_t and the decoder input x_t.

$$p_{gen} = \sigma\left(w_{h^*}^T h_t^* + w_s^T s_t + w_x^T x_t + b_{ptr}\right) \quad (4)$$

Where $w_{h^*}^T$, w_s^T, w_x^T and b_{ptr} are learnable parameters and σ is the sigmoid function. Then the p_{gen} is used as a soft switch to choose between generating a word from the vocabulary by sampling from p_{vocab} or copying a word from the input sequence by sampling from the attention distribution a^t .

$$P_{vocab} = \text{softmax}(V'(V[s_t, h_t^*] + b) + b') \quad (5)$$

$$P(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) \sum_{i:w_i=w} a_i^t \quad (6)$$

Note that if w is an out-of-vocabulary (OOV) word, then $p_{vocab}(w)$ is zero; similarly, if w does not appear in the source document, then $\sum_{i:w_i=w} a_i^t$ is zero. Their model has the ability to produce OOV words which is one of the primary advantages compared to the original Seq2Seq model.

3. NEURAL SUMMARIZATION MODELS

In this section, we describe the baseline sequence-to-sequence model, pointer-generator network and our two-encoder pointer-generator network models.

A. Sequence-to-sequence attentional model

We utilize the sequence-to-sequence attentional model whose input is only target segment text as our baseline

model. This Seq2Seq model is similar to that of Abigail et al. [2]. Figure 1 shows the structure of this model. The words of target segment w are fed into the encoder (one single-layer Bi-LSTM) one by one in encoding timestep i, producing a sequence of encoder hidden states h. Then the decoder (one single-layer LSTM) receives the word embedding of the word from the previous step (when training, the previous word is from the reference segment summary) on each step t and has decoder state s_t.

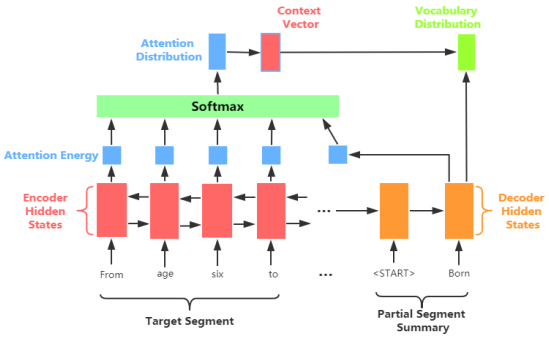

Fig. 1. sequence-to-sequence attentional model

Then the attention distribution a^T is calculated by Bahdanau attention [4] in (1) and (2). The attention distribution can be considered as a probability distribution over the source words, which helps the decoder decide where to focus to produce next word.

Next the attention distribution is used to produce a weighted sum of the encoder hidden states, context vector h_t^* in (3). The context vector can be viewed as a fixed-length representation of what has been read from the target segment for this step.

Then the context vector is concatenated with the decoder state s_t and fed through two linear layers to produce the vocabulary distribution p_{vocab} in (5). p_{vocab} is a probability distribution over all words in the vocabulary. In this model the final distribution $P(w)$ to predict words w is:

$$P(w) = P_{vocab}(\mathrm{w}) \quad (6)$$

The loss for training of decode step t is the negative log likehood of the target word w_t^* :

$$loss_t = -\log P(w_t^*) \quad (7)$$

The overall $loss$ for the whole sequence is:

$$loss = \frac{1}{T}\sum_{t=0}^{T} loss_t \quad (8)$$

The experiment of this model has two input settings. One is the baseline that uses only target segment and the other concatenates the target segment and compressed whole article as one input sequence to the model.

B. Pointer-generator network

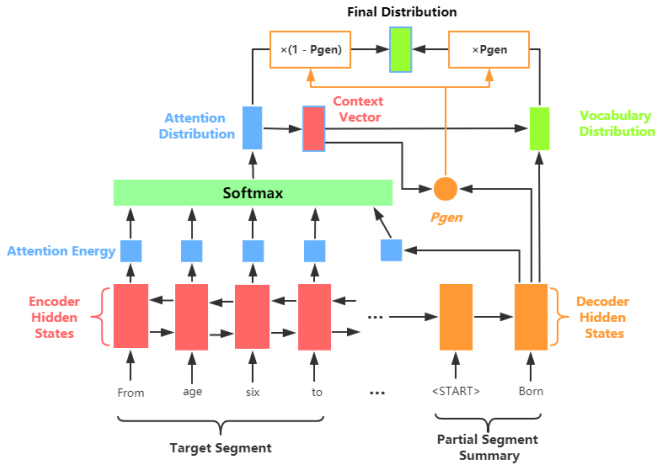

Fig. 2. Pointer-generator network

We utilized pointer-generator network with coverage mechanism proposed by Abigail et al. [2], which allows both copying words by pointing from the source text and generating words from a fixed vocabulary. Figure 3 shows the structure of this model.

For the purpose of solving the common repetition problem for sequence-to-sequence models, they proposed coverage mechanism. They defined the sum of attention distributions over all previous decoder timesteps as coverage vector c^t:

$$c^t = \sum_{t'=0}^{t-1} a^{t'} \quad (9)$$

The coverage vector c^t could be considered as a distribution over the source document words that represents the degree of coverage that those words have received from the attention mechanism so far. Note that c^0 is a zero vector because on the first decoding timestep, none has been covered from the source document. Then the coverage vector is used as extra input to the attention mechanism. Changing the formula to calculate the attention energy from (1) to:

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + W_c c_i^t + b_{attn}) \quad (10)$$

Where W_c is a learnable parameter vector of same length as v. Adding the coverage vector to the attention mechanism making the attention mechanism's current decision is considering its previous decisions which summarized in the coverage vector c^t.

Then the attention distribution a^t and context vector h_t^* are calculated as in (2) and (3). The generation probability $p_{gen} \in [0,1]$ for timestep t is calculated as in (4). Then the final probability distribution is calculated in

(6).

They also defined a coverage loss to penalize repeatedly attending to the same locations:

$$covloss_t = \sum_i \min(a_i^t, c_i^t) \qquad (11)$$

The coverage loss is bounded as: $covloss_t \leq \sum_i a_i^t = 1$. Then they reweighted the coverage loss by hyperparameter λ and added it to the primary loos function. The final loss function is:

$$loss_t = -\log P(w_t^*) + \lambda\, covloss_t \qquad (12)$$

For experiments, we use two input settings. One is using target segment text only and the other is using the concatenated sequence with target segment and additional sequence.

Considering just concatenating the target segment and additional information sequence might make it hard for decoder to recognize which part is from the segment. In addition, the encoder is using Bi-LSTM which means the additional sequence would affect the target segment's result. Considering the noise information in additional sequence might have bad influence, we also consider another model that uses different encoders to process the target segment and additional sequence separately.

C. Two-encoder Pointer-generator network

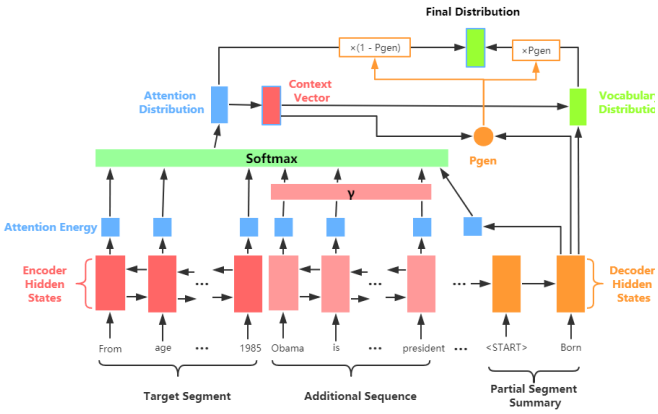

Fig. 3. Two-encoder Pointer-generator network

This model is similar to the previous model. There are two encoders in the model, one is to process the target segment and the other is to process the additional sequence. Figure 3 shows the structure of this model.

There are two hidden state sequences produced by two different encoders. The target segment encoder produces hidden states h^1, and the additional sequence encoder produces hidden states h^2. Having two hidden state sequences makes it not suitable for original Bahdanau attention mechanism. So we are changing the attention to fit our problem. We calculate the attention energy for two hidden state sequences separately:

$$e1_i^t = v1^T \tanh(W1_h h_i^1 + W1_s s_t + W1_c c_i^t + b1_{attn}) \qquad (13)$$

$$e2_i^t = v2^T \tanh(W2_h h_i^1 + W2_s s_t + W2_c c_i^t + b2_{attn}) \qquad (14)$$

where $e1_i^t$ is the target segment's attention energy and $e2_i^t$ is the additional sequence's attention energy. $v1^T$, $W1_h$, $W1_s$, $b1_{attn}$, $v2^T$, $W2_h$, $W2_s$ and $b2_{attn}$ are learnable parameters. Then we concatenate the two attention energy sequences into one by a learnable parameter γ (1.0 as initial value) to calculate the whole attention weight distribution a^t :

$$a^t = \text{softmax}([e1^t, \gamma e2^t]) \qquad (15)$$

Then we change the formula to calculate the context vector using attention weight distributions from (3) to:

$$h_t^* = \sum_{i=0}^{k} a_i^t h_i^1 + \sum_{i=k+1}^{m+k} a_i^t h_i^2 \qquad (16)$$

where k is the length of the target segment encoder's hidden states sequence and m is the length of the additional sequence encoder's hidden state sequence.

Then the remaining step is similar to the previous pointer-Generator network model. We use the same calculation to obtain the final distribution, the loss is still using the formula (12).

4. TECHNIQUES FOR ADDITIONAL SEQUENCE

In this paper, we consider that the contents outside the target segment in the article contains useful information for segment summarization. But normally the whole article would be too long. Such long sequences would make it hard to train the RNN-based model. In addition, we cannot ignore that there would also be a large volume of noise information outside the target segment. So in this section, we discussed three single-document extractive method to extract the important sentences from the article to form the additional input sequence.

A. Basic method– leading sentences

The basic method works by picking leading two sentences in each segment of the article and concatenate these sentences into one sequence as the additional sequence for our model.

According to Abigail et al. [2], their experiment using only leading three sentences performs better than using longer inputs. We consider that the important information is usually in the leading sentences. Thus, we utilize this method as our basic method to extract important sentences from the target article.

B. TextRank + word2vec

TextRank is a graph-based ranking algorithm inspired by PageRank [16] algorithm. It was first proposed by Mihalcea and Tarau [12]. TextRank is mainly used for keyphrase extraction and important sentence extraction. In our work, we are using it for sentence extraction.

To apply TextRank and word2vec on sentence extraction, they first build a graph associated with the text, where the graph vertices are representative for the sentences in the document to be ranked. The main steps are:

(1) Split the target document into sentences $V = [S_1, S_2, S_3, \dots, S_m]$, and construct graph $G = (V, E)$. Then split sentences into words and remove stop words to obtain $S_i = [t_{i,1}, t_{i,2}, \dots, t_{i,n}]$, where $t_{i,j} \in S_i$ is the candidate keyword.

(2) Using word2vec to convert words in a sentence to vectors and calculate the average vector of the sentence:

$$vec(S_i) = \frac{1}{n}\sum_{j=1}^{n} word2vec(t_{i,j}) \quad (17)$$

(3) Calculate the cosine similarity between sentences as the edge weights in graph G:

$$Similarity(S_i, S_j) = cos(vec(S_i), vec(S_j)) \quad (18)$$

(4) Calculate the score of each sentence, where the score of S_i is:

$$WS(S_i) = (1-d) + d\sum_{S_j \in In(S_i)} \frac{w_{ji}}{\sum_{S_k \in Out(S_j)} w_{jk}} WS(S_j) \quad (19)$$

(5) Pick the top-10 highly scored sentences and sort by the positions of sentences in the document. Concatenate the sentences as our additional sequence.

C. TextTeaser

TextTeaser is a feature-based extractive summarization algorithm. It selects sentences that possess best score among others. Its' features include title feature, sentence length, sentence position and keyword frequency.

5. EXPERIMENTS

A. Datasets

We collected 26,318 pages from English version of Wikipedia as our dataset for the experiments. There are 81,124 segments and 50,056 sections altogether in the dataset. We convert all the words to lower case and the numbers to "TAG_OF_NUM".

Since there is no human labeled segment summary in Wikipedia articles, and writing this number of segment summaries is not realistic. So, we utilized Word2Vec to calculate the average vector of segment title and segment sentences. Then use cosine similarity to compare segment title and segment sentences and choose the most similar sentence. We concatenate it after the segment title as our golden standard.

B. Settings

For all experiments, we follow the settings in Abigail et al. [2]. Our models have 256-dimensional hidden states and 128-dimensional word embeddings. For the pointer-generator network models, we use a vocabulary of 50k words for both source and target.

The word embeddings we are using are not pre-trained. They are learned from scratch during training. We train using Adagrad [10] with learning rate 0.15 and an initial accumulator value of 0.1. We use gradient clipping with a maximum gradient norm of 2 without using any form of regularization.

For training and testing, we truncated the target segment and additional sequence into leading 400 words and limit the length of the summary to 50 words. According to Abigail et al. [2], truncating the input sequence could raise the performance of the model.

We trained our models on a single 1080Ti GPU with a batch size of 32. For testing, we utilized beam search algorithm to produce segment summaries with beam size 8.

To compare the performance between models, we trained our models separately using the additional sequence which is generated by the additional techniques including leading sentences, TextRank and TextTeaser. We also did the experiment with different length of target segments to see how the length affect the result.

C. Results

Our results are shown in Table I. We evaluate our models with the standard ROUGE metric [3], reporting the F_1 scores for ROUGE-1, ROUGE-2 and ROUGE-L, which respectively measure the word-overlap, bigram-overlap, and longest common sequence between the reference summary and the summary to be evaluated. According to Abigail et al. [2], ROUGE trends to reward safe strategies such as selecting the first-appearing content, or preserving original phrasing, which may lead to extractive systems obtains higher ROUGE scores on average. Therefore, we also evaluate our systems with the METEOR metric, which rewards not only exact word matches, but also matching stems, synonyms and paraphrases.

TABLE I. RESULTS OF DIFFERENT MODELS

Models	Additional Sequence Method	ROUGE			METEOR	
		1	2	L	Exact	+stem/syn/para
TextRank	None	44.99	33.20	38.84	26.72	27.46
Seq2Seq attentional model	None	36.70	19.33	29.90	17.08	17.65
Pointer-Generator Network	None	42.17	28.01	35.82	23.26	23.91
	Basic (leading)	42.70	29.02	36.00	25.20	25.73
	TextRank	43.96	30.23	37.31	26.33	26.86
	TextTeaser	43.13	29.43	36.38	25.46	25.98
Two-Encoder Pointer-Generator Network	TextRank	44.69	31.04	38.17	27.17	27.69

For the baseline models, the Seq2Seq attentional model and Pointer-Generator network, applying additional sequence method means concatenating the additional sequence after the target segment. For no-additional sequence method, only truncated target segment was input to the model.

In the results of the no-additional sequence method, we can see that pointer-generator network outperforms the traditional Seq2Seq attentional model. The model trains faster and takes less iterations. Thus, in succeeding experiments on additional sequence methods we only focus on Pointer-Generator network.

We first perform experiments that simply concatenate the additional sequence after the target segment. The results show that our idea of additional sequences from the entire article is improving the performance. Among them, sequences that extracted by TextRank perform the best. Therefore, for our models that process two sequences separately, we use additional sequences extracted by TextRank only.

The last row of Table 1 is the result of our proposing model Two-Encoder Pointer-Generator network. As we can see from the scores, our model out-performed all the other models including original Pointer-Generator network with additional sequences by TextRank. Thus, our model that uses two encoders to process the target segment and the additional sequence separately appears to be working.

We also evaluate TextRank as an extractive method to extract summary from the target segment. The first row shows TextRank is slightly better than our model. We consider the possible reason can be that the major part of golden standard in our dataset is one sentence from the target segment.

Additionally, we performed experiments on how the length of a target segment affects the result. We divided the testing dataset into three parts by the length of the target segments: less than 200 words; between 200 and 400 words; over 400 words. Then we use our trained Two-Encoder Pointer-Generator network model from the last experiment to produce the results. Table II shows the results.

TABLE II. EXPERIMENT ON LENGTH OF TARGET SEGMENTS

Length	ROUGE			METEOR	
	1	2	L	Exact	+stem/syn/para
Overall	44.69	31.04	38.17	27.17	27.69
(0,200]	53.86	40.65	47.30	33.83	34.26
(200,400]	45.39	33.19	40.70	28.73	28.93
$(400,\infty)$	29.86	15.65	23.68	16.53	17.20

The results show that our model performs much better when the target segment is less than 200 words and slightly better between 200 and 400 words. But it becomes worse when the length exceeds 400 words. It implies that our model is more capable to handle shorter articles, which is one of the existing common problems for RNNs that longer articles are harder to process, including LSTM. Also, shorter segments mean a smaller number of sentences, given a higher possibility to locate the reference sentence in the segment.

Considering that there has no perfect evaluation method for text summarization, we picked several results from the four different methods we evaluated and observed by human eyes. Table III is two of those examples. We can see that the Seq2Seq model are clearly the worst model. It failed to generate a large number of Out-Of-Vocabulary (OOV) words, which the Pointer-Generator Network is capable of.

TABLE I. SUMMARY EXAMPLES

Reference summary: **TAG_OF_NUM state representative election in TAG_OF_NUM.** newhall was elected as a democrat to serve in the massachusetts house of representatives of TAG_OF_NUM.	**TAG_OF_NUM.** on march TAG_OF_NUM, TAG_OF_NUM, johnston released his first comic book published by boom!.
Seq2Seq Attention model: state representative in the TAG_OF_NUM election [UNK] received TAG_OF_NUM [UNK] TAG_OF_NUM votes coming in second behind republican candidate john [UNK] gonzalez who received TAG_OF_NUM [UNK] TAG_OF_NUM [UNK] and just ahead of fellow democrat michael [UNK] jay who received TAG_OF_NUM [UNK] TAG_OF_NUM [UNK]	and jonathan [UNK] on march TAG_OF_NUM [UNK] TAG_OF_NUM [UNK] johnston released his first comic [UNK] at [UNK] published by [UNK] [UNK] [UNK] comic book [UNK] with the album and an warren's [UNK] has collaborated with skateboarding and clothing company supreme on numerous collections [UNK] of clothing and various [UNK].
Pointer-Generator Network Without Additional Sequence: TAG_OF_NUM election in the TAG_OF_NUM election newhall received TAG_OF_NUM , TAG_OF_NUM votes coming in second behind republican candidate john w. blaney who received TAG_OF_NUM , TAG_OF_NUM votes, and just ahead of fellow democrat michael f. phelan who received TAG_OF_NUM , TAG_OF_NUM [UNK]	on march TAG_OF_NUM , TAG_OF_NUM , johnston released his comic book, published by boom! [UNK] the comic book ties-in with the album and an ios app. johnston has collaborated with skateboarding and clothing company supreme on numerous collections
Pointer-Generator Network With TextRank additional Sequence: congress in TAG_OF_NUM newhall was elected as a democrat to serve in the massachusetts house of representatives of TAG_OF_NUM.	and on march TAG_OF_NUM , TAG_OF_NUM , johnston released his first comic book, at sxsw, published by boom! studios.
Two-Encoder Pointer-Generator Network (proposed): TAG_OF_NUM election in TAG_OF_NUM newhall was elected as a democrat to serve in the massachusetts house of representatives of TAG_OF_NUM.	TAG_OF_NUM on march TAG_OF_NUM , TAG_OF_NUM , johnston released his first comic book, at sxsw, published by boom! studios.
TextRank: newhall was elected as a democrat to serve in the massachusetts house of representatives of TAG_OF_NUM.	on march TAG_OF_NUM, TAG_OF_NUM, johnston released his first comic book published by boom!

• ***Green*** *denotes segment title in the reference summary*

Furthermore, Both the Seq2Seq attentional model and Pointer-Generator Network are generating a large number of extra words we need. Also, our methods with additional sequence are not generating so much words. This maybe because that the additional sequence we are using is extracted from the entire article including the target segment. By adding the additional sequence as an extra input to the model, the most important information of the target segment is emphasized twice

6. DISSCUSSION

A. How the additional sequence works

It is clear from Table I that applying additional sequence tends to achieve higher scores than none, no matter simple concatenation or using two encoders to process it separately. We offer two possible explanations for these observations.

Firstly, for Wikipedia articles that have multiple segments, segments are usually not isolated from each other. They tend to contain similar information. It is possible that these information could be extracted into the additional sequence that would be useful for segment summarization.

Secondly, the additional sequence can be regarded as compressed information from the entire article. It is likely that it contains the target segment's most important information. By supplying the target segment and the additional sequence, the important information in the target segment could be emphasized.

Furthermore, we consider that additional sequence could be useful for generating longer summaries like news article. Existing similar news article's summary could be used as e additional sequences to the model, since the summary of similar articles usually have similar writing

patterns. By introducing similar news articles' highly compressed summary, words that widely used in summary but not appearing in the article could be introduced, which could help the system more easily to generate.

B. Why processing separately performs better

Rows 5 and 7 in Table I show that using the same additional sequence and target segment, processing them separately scores higher than concatenation. We have several explanations for that.

As we can see in Table II, the results are getting worse when the target segment is getting longer. We consider this as the common problem for RNNs that they have problems handling longer articles. Therefore, in our problem, concatenating the target segment and additional sequence is making the input sequence longer. Which would make it harder for our LSTM-structured encoder to handle. Thus, in our two-encoder model, using two LSTMs to process the target segment and additional sequence can avoid this problem in some degree.

Using several LSTMs to process parts of the input text has been researched by researchers. Tan et al. [11] are using a hierarchical encoder-decoder framework for text summarization. Tan utilized word encoders and sentence encoders in their model and it shows effectiveness. Thus, we can assume that in future study, we could use a similar structured hierarchical encoder-decoder model to process long articles, by dividing the encoders into word encoders, sentence encoders and maybe paragraph encoders.

7. CONCLUSIONS

In this paper, we performed experiments on three different models for Wikipedia article segment summarization work including Seq2Seq + attention, Pointer-Generator network and our Two-Encoder Pointer-Generator network. We also considered three different ways to extract important information from the whole article. The results show that our idea of additional sequence and processing it separately is effective. Our models out-performed the other models. Although there still exist problems, one is that our model suffers from longer segments. But our Two-Encoder model's results suggest that the LSTM has the potential for dealing with longer articles, if an appropriate hierarchical structure is employed.

References

[1] Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. "A neural attention model for abstractive sentence summarization". In Empirical Methods in Natural Language Processing.

[2] A See, PJ Liu, CD Manning. 2017. "Get To the Point: Summarization with Pointer-Generator Networks". Annual Meeting of the Association for Computational Linguistics

[3] Chin-Yew Lin. 2004a. Looking for a few good metrics: Automatic summarization evaluation-how many samples are enough? In NACSIS/NII Test Collection for Information Retrieval (NTCIR) Workshop

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. "Neural machine translation by jointly learning to align and translate". In Computer Science.

[5] Dat P.T Nguyen, Yutaka Matsuo, and Mitsuru Ishizuka. "Exploiting Syntactic and Semantic Information for Relation Extraction from Wikipedia" Text-Mining and Link-Analysis (TextLink2007), 2007.

[6] Gers, Felix A., Nicol N. Schraudolph, and Jürgen Schmidhuber. "Learning precise timing with LSTM recurrent networks." Journal of machine learning research 3, no. Aug (2002): 115-143.

[7] Graves and Schmidhuber, J. "Framewise phoneme classification with bidirectional LSTM networks". In IEEE International Joint Conference on Neural Networks. 2005, 4:2047-2052

[8] Hu M, Sun A and Lim E-p 2007 Comments-oriented Blog Summarization by Sentence Extraction Proc. of the Sixteenth ACM Conf. on Conf. on Information and Knowledge Management 901-4 (Lisbon, ACM)

[9] Ilya Sutskever, Oriol Vinyals, Quoc V. Le. Sequence to Sequence Learning with Neural Networks. In NIPS, 2014. 2, 3, 7

[10] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research 12:2121–2159.

[11] Jiwei Tan, Xiaojun Wan and Jianguo Xiao. 2017. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1171-1181

[12] Mihalcea R, Tarau P. TextRank: Bringing order into text. In Proceedings of the 2004 conference on empirical methods in natural language processing 2004

[13] M. Volkel, M. Krotzsch, D. Vrandecic, H. Haller, and R. Studer. Semantic Wikipedia. In Proceedings of the WWW2006, pages 585-594, 2006.

[14] Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. "Multi-task sequence to sequence learning". In ICLR.

[15] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. "Pointer networks". In Neural Information Processing Systems.

[16] Page, Lawrence, et al. The PageRank citation ranking: Bringing order to the web. Stanford Info Lab, 1999.

[17] Rada Mihalcea and Paul Tarau. "TextRank: Bringing Order into Texts" 2004

[18] S Hochreiter, J Schmidhuber. 1997. "Long short-term memory". In Neural Computation, 1997, 9(8):1735-1780

[19] Vishal Gupta, Gurpreet Singh Lehal. "A Survey of Text Summarization Extractive Techniques". Journal of emerging technologies in web intelligence (Vol. 2, No. 3, August 2010).