NEWS HEADLINE-CONTENT GENERATOR: BUILDING SEQ2SEQ ARCHITECTURE USING NUMPY FROM SCRATCH

A PREPRINT

Hao-Chien, Hung
Institute of Atomic and Molecular Sciences, Academia Sinica
Taipei, 10617, Taiwan
hchungdelta@gmail.com

February 20, 2019

ABSTRACT

The whole machine learning model in this paper is writed using Numpy, which is available on Github (see footnote). The basic LSTM seq2seq architecture is adopted in this model, the self-attention mechanism and the layer normalization are applied to enhance the performance and the training efficiency. Furthermore, parallel computing is implemented to speed up training. In this paper, the main task is to test whether the model can be used for news headline-content generation. Only 306k news headline-content data are used for training, however, even with relatively small data, the results are reasonably good and indicate the promising future for further developing.
[1]

***K*eywords** Natural Language Processing · Deep Learning · Attention Mechanism · Layer Normalization

1 Introduction

Recently, there is an unprecedented progress in natural language processing (NLP), which attributes to the rapidly developing neural network architecture and the computing power increases in the order of magnitude just within these few decades. The dominant learning model in NLP is the encoder-decoder neural network, which is known as seq2seq model, has a multifaceted capability to perform a variety of tasks, such as machine translation, speech recognition, and text generation.

Nowadays, most of the seq2seq models are using long-short-term memory(LSTM)[1] or gated recurrent unit (GRU)[2] as the basic building block, which is less likely to suffer from loss of information compared to the traditional recurrent neural network(RNN). Subsequently, the introduction of attention mechanisms [3, 4] further boost the performance of seq2seq model. More recently, a new attention mechanism called self-attention have been introduced by Ashish Vaswani *et al.* [5], which a set of key-value pairs are adopted. In this paper, a different version of self-attention is implemented, which will be discussed later.

To train the LSTM seq2seq model is computationally expensive, and it potentially suffers from internal covariate shift as the neural network become deeper, which further slow down the training. Fortunately, layer normalization [6] has been introduced to efficiently solve this problem. Layer normalization normalizes the activities of the neurons and hence mitigates the risk of vanishing gradient problem during training. Compare to batch normalization [7], layer normalization doesn't require a large amount of batch during training and doesn't need to deal with different input sequence length. Therefore, layer normalization is generally considered more suitable to be combined with LSTM-based model rather than batch normalization.

In this paper, the main purpose is to test whether the seq2seq model can be used as a news headline-content generator. That is, input the headline, the model can generate the content only dependent on the headline we input. The paper is

[1] Code is available on Github `https://github.com/hchungdelta/Simple_NN_API/tree/master/NN_v.2.1_news_generator`

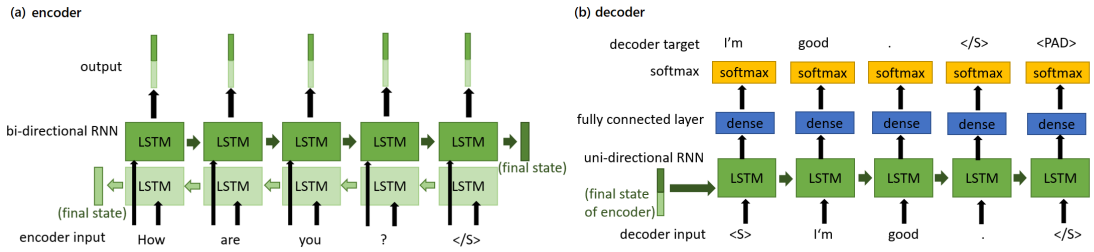

Figure 1: The schematic diagrams of seq2seq model. (a) The encoder, where the Bi-RNN mechanism is used. (b) The decoder, where the uni-RNN mechanism is used. <S>, </S>, and <PAD> represent the start token, end token, and padding token, respectively.

organized as follows. Section 2 presents the architecture and the basic algorithm of this model. In section 3, the results and discussions are presented. Summary and conclusion are included in section 4.

2 Neural Network Architecture

2.1 LSTM

In this paper, the LSTM mechanism [1] is adopted as the basic building blocks of my seq2seq model. The LSTM mechanism is shown below:

$$forget\ gate : hf_t = \sigma(W_{fh}h_{t-1} + W_{fx}x_t + b_f) \tag{1}$$
$$input\ gate : hi_t = \sigma(W_{ih}h_{t-1} + W_{ix}x_t + b_i) \tag{2}$$
$$output\ gate : ho_t = \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_o) \tag{3}$$
$$candidate\ output : hc_t = tanh(W_{oc}h_{t-1} + W_{cx}x_t + b_c) \tag{4}$$
$$hidden\ state : c_t = hf_t * c_{t-1} + hi_t * hc_t \tag{5}$$
$$hidden\ output\ h = ho_t * tanh(c_t) \tag{6}$$

A number of LSTM form the neural network sequence. In seq2seq model, two sequences are needed, which are known as the encoder and the decoder.

2.2 Seq2Seq Model

For the encoder, the Bi-directional RNN [8] is implemented to process the inputs. For the decoder, since future inputs aren't available, uni-directional RNN is used. A simple schematic diagram is depicted in Fig. 1.

2.3 Layer Normalization

Layer normalization [6] is implemented to improve the efficiency of training. In the early stage, batch normalization [7] is also adopted to train the model. However, the results indicate that it potentially makes the model less stable, since the lengths of input data are usually different, which make batch normalization less suitable for RNN. Therefore, only layer normalization is used in this paper. The mechanism of layer normalization is shown in the following:

$$mean : \mu_d = \frac{1}{d}\sum_{i=0}^{d} x_i \tag{7}$$
$$variance : \sigma_d^2 = \frac{1}{d}\sum_{i=0}^{d} (x_i - \mu_d)^2 \tag{8}$$
$$normalization : \hat{x} = \frac{x - \mu_d}{\sqrt{\sigma_d^2 + \epsilon}} \tag{9}$$
$$scale/shift : LN(x) = \gamma\hat{x} + \beta \tag{10}$$

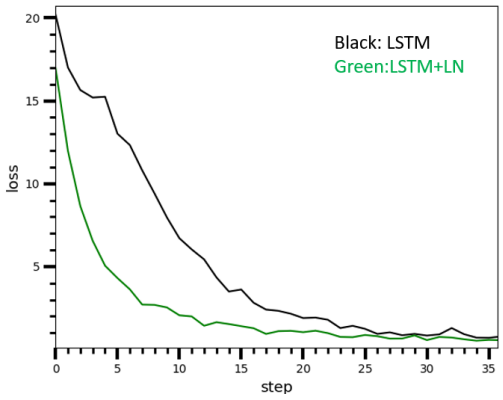

Figure 2: The comparison of the training efficiency of pure LSTM model and LSTM model with layer normalization.

d represents the depth of the input. i in x_i represents the $i-th$ scalar value in this input. γ and β are both vectors to re-scale the input. As the weights and the biases in LSTM, the parameters of these two vectors are shared among the RNN sequence. The backpropagation is similar to those used in batch normalization [7]. As the following expresses:

$$\frac{\partial L}{\partial \gamma} = \sum_{i=0}^{d} \frac{\partial L}{\partial LN(x)} \cdot \hat{x} \tag{11}$$

$$\frac{\partial L}{\partial \beta} = \sum_{i=0}^{d} \frac{\partial L}{\partial LN(x)} \tag{12}$$

$$\frac{\partial L}{\partial \hat{x}} = \frac{\partial L}{\partial LN(x)} \cdot \gamma \tag{13}$$

$$\frac{\partial L}{\partial \sigma_d^2} = \sum_{i=0}^{d} \frac{\partial L}{\partial \hat{x}} \cdot (x - \mu_d) \cdot \frac{-1}{2} (\sigma_d^2 + \epsilon)^{\frac{-3}{2}} \tag{14}$$

$$\frac{\partial L}{\partial \mu_d} = \sum_{i=0}^{d} \frac{\partial L}{\partial \hat{x}} \cdot \frac{-1}{\sqrt{\sigma_d^2 + \epsilon}} + \frac{\partial L}{\partial \sigma_d^2} \cdot \frac{\sum_{i=0}^{d} -2(x - \mu_d)}{d} \tag{15}$$

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial \hat{x}} \cdot \frac{1}{\sqrt{\sigma_d^2 + \epsilon}} + \frac{\partial L}{\partial \sigma_d^2} \cdot \frac{2(x - \mu_d)}{d} + \frac{\partial L}{\partial \mu_d} \cdot \frac{1}{d} \tag{16}$$

After applying layer normalization, the LSTM mechanism now should be rewritten as:

$$hf_t = \sigma(LN_{fh}(W_{fh} h_{t-1}) + LN_{fx}(W_{fx} x_t) + b_f) \tag{17}$$

$$hi_t = \sigma(LN_{ih}(W_{ih} h_{t-1}) + LN_{ix}(W_{ix} x_t) + b_i) \tag{18}$$

$$ho_t = \sigma(LN_{oh}(W_{oh} h_{t-1}) + LN_{ox}(W_{ox} x_t) + b_o) \tag{19}$$

$$hc_t = tanh(LN_{ch}(W_{oc} h_{t-1}) + LN_{cx}(W_{cx} x_t) + b_c) \tag{20}$$

$$c_t = hf_t * c_{t-1} + hi_t * hc_t \tag{21}$$

$$h = ho_t * tanh(LN_o(c_t)) \tag{22}$$

The comparison of training efficiency of the model with and without layer normalization is displayed in Fig. 2.

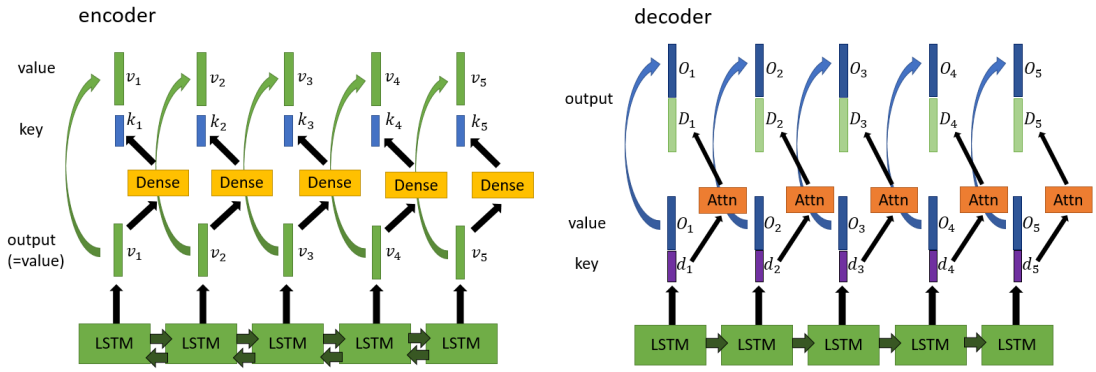

Figure 3: The schematic diagram of the attention mechanism implemented in this paper

2.4 Attention mechanism

Dot-product attention mechanism [5] is adopted in this paper, although a little change in algorithm is made. As illustrated in 3. The encoder output) and the decoder output are separated into key parts (k_i for encoder key, d_i for decoder key) and value parts (v_i for the encoder value, O_i for the decoder value). Afterward, the original decoder value and the attention output are concatenated as the final decoder output.

The algorithm is as follows:

$$score_{ij} = s_{ij} = d_i \cdot k_j \tag{23}$$

$$alpha_{ij} = \alpha_{ij} = \frac{e^{s_{ij}}}{\sum_{x=0}^{x_{max}} e^{s_{ix}}} \tag{24}$$

$$attention\ output_i = D_i = \sum_{x=0}^{x_{max}} \alpha_{ix} v_x \tag{25}$$

$$output_i = concat(O_i, D_i) \tag{26}$$

For the backpropagation:

$$\frac{\partial L_i}{\partial v_j} = \frac{\partial L_i}{\partial D_i}\frac{\partial D_i}{\partial v_j} = \frac{\partial L_i}{\partial D_i}\alpha_{ij} \tag{27}$$

$$\frac{\partial L_i}{\partial d_i} = \frac{\partial L_i}{\partial D_i}\frac{\partial D_i}{\partial d_i} = \frac{\partial L_i}{\partial D_i}\sum_{x=0}^{x_{max}}\frac{\partial}{\partial d_i}\alpha_{ix}v_x = \frac{\partial L_i}{\partial D_i}[\sum_{x=0}^{x_{max}}\alpha_{ix}k_x v_x - \sum_{x=0}^{x_{max}}\sum_{y=0}^{x_{max}}\alpha_{ix}\alpha_{iy}k_y v_x] \tag{28}$$

$$\frac{\partial L_i}{\partial k_j} = \frac{\partial L_i}{\partial D_i}\frac{\partial D_i}{\partial k_j} = \frac{\partial L_i}{\partial D_i}\frac{\partial}{\partial k_j}\sum_{x=0}^{x_{max}}\alpha_{ix}k_x v_x = \frac{\partial L_i}{\partial D_i}[\delta_{ij}d_i v_j - \alpha_{ix}\alpha_{ij}d_i v_x] \tag{29}$$

In this model, since the encoder training data is much shorter than the decoder, only a single-headed attention mechanism is implemented, rather than multi-headed attention mechanism. Fig. 4 compares the efficiency of the model with and without attention mechanism.

2.5 Orthogonal initialization

Rather than randomly generate the weight matrices as initial states, in this paper, the weight matrices are generated using the concept called orthogonal initialization. It can potentially help the learning model recognize the difference between each input.

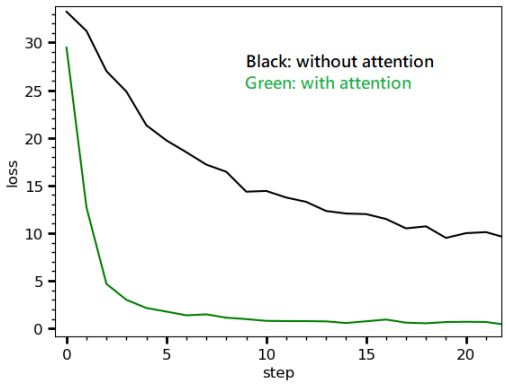

Figure 4: The comparison of the training efficiency of the model with and without attention mechanism.

2.6 Parallel Computing

Parallel computing is implemented by mpi4py. In current development, only embarrassing parallel computing is available in this code, since it is effortless to apply to all the learning models without further adjustments. All the nodes share the same weights (W) and biases (b) throughout the training while exposing to different training data. For instance, if four nodes are used for training and the batch is set to be 32, training data no.1 to no.32 will be delivered to node one, no.33 to no.64 will be delivered to node two, and so on. Hence at each step, the total amount of 128 training data are used for training. Afterward, the master node will sum up all the gradients of weights (dW) and biases (db) to optimize the model. As the following equations show.

$$dW_{sum} = \sum_{node=1}^{node_{max}} dW_{node} \tag{30}$$

$$db_{sum} = \sum_{node=1}^{node_{max}} db_{node} \tag{31}$$

The summed gradients of weights and biases will be sent to the optimizer for updating the original weights and biases (See Sec. 2.7).

The schematic diagram (Fig. 5 (a)) visualizes the notion of the embarrassing parallel computing in this code. If one wants to take an average of the gradients to avoid overshooting, just simply divided the learning rate by the amount of the nodes. Fig. 5 (b) depicts the loss-step curves for the same learning model while using a different amount of nodes during training. The plot clearly indicates the parallel computing is useful and efficient. Until now, there is no dramatic deterioration in accuracy found when using parallel computing.

2.7 Optimizer

Stochastic Gradient Descent (SGD), Momentum, and Adaptive Moment Estimation (Adam) optimizer [9] are available in this code. In this paper, only Adam optimizer is used for training since the efficiency of Adam can outperform the other two approaches nearly in the order of magnitude. The learning rate (lr) is set to be 0.0015. ϵ, β_1, and β_2 are equal to $1e^{-8}$, 0.9, and 0.999, respectively. The following shows how the Adam optimizer works, t represents the step during training (here only take the gradients of the weights for example, the same goes for biases).

$$dW_{t+1} = dW_t - lr\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{32}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}\ ,\ \hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{33}$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)dW_t \tag{34}$$

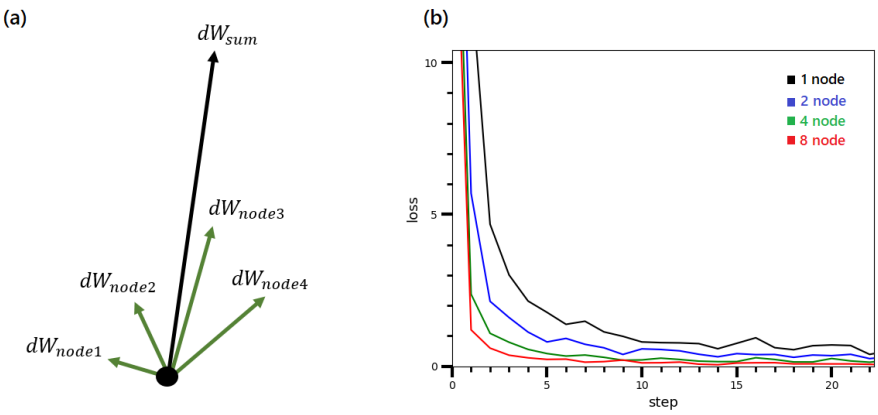

Figure 5: (a) The schematic diagram of embarrassing parallel computing implemented in this model. (b) The comparison of the loss-step curves of the same learning model trained with different amount of nodes.

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) dW_t^2 \tag{35}$$

The gradient clipping [10] is also implemented to help model converge more smoothly. In this paper, the threshold is set to be 1.5. The equation is shown below:

$$dW_{ij} = \begin{cases} threshold, & \text{if } dW_{ij} > threshold \\ dW_{ij}, & \text{otherwise} \\ -threshold, & \text{if } dW_{ij} < -threshold \end{cases} \tag{36}$$

2.8 Word2vec

The word vector has been proved to be more efficient, and serve as a better representation for the words compared to the simple one-hot vector. First, jieba is applied to separate the Chinese sentence into vocabularies, and the digits are separated one by one (for instance, 2010 will be expressed as 2, 0, 1, and 0). Afterward, word2vec(using genism) is implemented to generate the word vector. To avoid including too many vocabularies in the dictionary, the threshold is set to be 50 (the lower limit of the number of occurrences), below which the vocabularies are labeled as "unknown". The depth of the word vector is 300 units. A few hundred MB news data was provided to train the word vector, as it turns out that nearly 69k vocabularies are over the threshold and be included in the dictionary.

2.9 Model

The neural network architecture used in this paper is illustrated in Fig. 6. The hidden units in LSTM are set to be 800 (for biRNN it is 1600 in total). The attention depth is 300 (depth of k_i and d_i). The input length (headline) and output length (content) are 24 and 90, respectively. The batch size is 32 during training. With only 1 CPU, it cost nearly a month to train the model. The results and discussions are detailed in the next section.

3 Results and Discussions

Unlike machine translation, there is no universal standard to judge the performance of news content-generation machine learning model. The evaluation of the performance of this model must be based on human judgment. In this section, two tests are carried out. The first test is to create a set of headlines, and then feed into the model to generate content, the evaluation is based on the performance of these generated results. The second test is to use recent news headlines as input (these data are not in training data) and check the consistency of our results and the real content of these news. Two criteria are considered to judge the performance of the model, the first is the grammar of the sentence, whether the sentence is grammarly correct or not. The second criterion is the consistency of the content and the headline, to see whether the model can generate relative content based on the input headline.

For the first test, the results are shown in Fig. 7. I create a few headlines(all of them are sarcastic) and feed them into the model. One can see that the result is quite reasonable. The model can understand the input and tries to generate

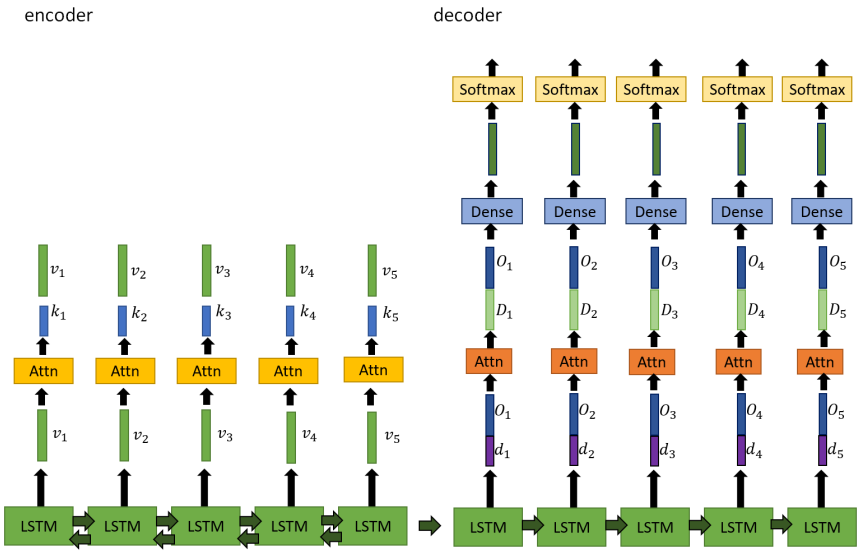

Figure 6: The neural network architecture used in this paper.

content based on it. Of course, the model absolutely needs to be further improved, too many unknown tokens appears, and the sentence is not so grammarly correct. Those problems will be discussed later.

For the second test, the results are shown in Fig. 8 and Fig. 9. Although the generated content is not consistent with real content (as expected), these generated content are still reasonable and interesting. If we modify these content, such as replace the unknown tokens with certain relative words and correct the grammar errors, this model is possible to reach the same level of performance as the journalists.

To check whether the attention mechanism works, the attention weight matrix alpha is visualized in Fig. 10. Obviously, the relative words have larger weights.(桃園:桃園/機場, 機場:機場, 水上:水上, 樂園:樂園). However, since the amount of training data is inadequate, some words in the content might pay attention to a non-relative term.

To enhance the performance, we need to improve the model from two perspectives. From algorithm, and from training data. From the algorithm perspective, one important issue is the unknown vocabulary, of course, includes all the vocabularies in the training data is inefficient. Perhaps the best solution is to adopt a word and character-hybrid learning model [11]. From the training data perspective, there are some problems need to be solved to enhance the performance of the model. (1) date & location (2) different publishers (3) new information (4) the draft of the news.

- (1) date & location: The date when the news published is also crucial. For instance, the president will change over a certain period. The same goes to the location, if the mayor is mentioned in the headline, it would be ambiguous that who we are referred to. Since we don't provide the information such as in which city we live in, it is highly possible that the model will generate content that is less relevant to our expectation. To improve consistency, the date & location are needed to be embedded in the model.

- (2) different publishers: What is intriguing is that, even with the same headline, different news publishers can generate totally different content depending on their political tendency to a certain party. The persona embedding mechanism can be applied to solve this issue.

- (3) new information: To understand a new statement is also an important issue. For example, if the news used as training data are published before 2010, it is unlikely that the model can recognize the term "machine learning".

- (4) the draft of the news: Perhaps to include the draft of the news, such as interview and some other information, can drastically boost the performance. Since the draft is highly relative to the content, and it is the raw material used to construct the news.

4 Conclusion

In this paper, a news headline-content generation seq2seq model is introduced. Although the training data are relatively small (about tens of MB) compare to the state-of-the-art seq2seq model, the results indicate the output content is quite reasonable. It guarantees news headline-content to be a fruitful field for further investigation and research. Although some challenges still remain as mentioned in 3, if we adopted a better machine learning algorithm and much more data for training, the performance, and the effectiveness are expectable. The whole machine learning algorithm is written in Numpy, it is available on Github.

`https://github.com/hchungdelta/Simple_NN_API/tree/master/NN_v.2.1_news_generator`

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. Long-short term memory. pages 1735–1780. Neural Computation 9(8), 1997.

[2] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoderapproaches. In *eprint arXiv:1409.1259*, 2014.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *eprint arXiv:1409.0473*, 2014.

[4] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *eprint arXiv:1508.04025*, 2015.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Llion Jones Jakob Uszkoreit, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *eprint arXiv:1706.03762v5*, 2017.

[6] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. In *eprint arXiv:1607.06450*, 2016.

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *eprint arXiv:1502.03167*, 2015.

[8] Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. volume 45. IEEE TRANSACTION ON SIGNAL PROCESSING 11, 1997.

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *eprint arXiv:1412.6980*, 2014.

[10] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *eprint arXiv:1211.5063*, 2012.

[11] Minh-Thang Luong and Christopher D. Manning. Achieving open vocabulary neural machine translation with hybrid word-character models. In *eprint arXiv:1604.00788*, 2016.

```
headline (original):  韓國瑜:復興禿頭風潮推廣高雄特色禿頭產業
headline (input):  韓國瑜:復興禿頭風潮推廣高雄特色禿頭產業</S>
generated context (output):
 國民黨高雄市長參選人韓國瑜今天(20日)一同政見, 以前可以說是年輕人的年輕人,
可以打李佳芬的就要找出他的選票, 是他推動的「特色產業」, 將以他的政見特色產業, 也創造出在高雄特色音樂實力
。</S>
================================================================================
headline (original):  桃園機場擬建立水上樂園
headline (input):  桃園機場擬建立水上樂園</S>
generated context (output):
桃園機場<UNK>, 桃園市政府今天啟動「第二<UNK>水上樂園站」, 6月29日在高雄巨蛋體驗<UNK><UNK>主題,
屆時也會建立公園站急<UNK>體驗水上樂園。</S>
================================================================================
headline (original):  學校申請不上恐導致學生無法入學
headline (input):  學校申請不上恐導致學生無法入學</S>
generated context (output):
為解決學生申請入學管道, <UNK><UNK>教育部長, 今天在<UNK>4<UNK>於學生宿舍申請不說,
已違反規定, 且不只要<UNK>學生, 不能參加學校考試, 讓學生入學也能向教育部申請。</S>
================================================================================
headline (original):  如何有效減肥又省錢?專家:別吃就好
headline (input):  如何有效減肥又省錢?專家:別吃就好</S>
generated context (output):
近日, 不少人認為身材的<UNK>受<UNK>為影響孩子的疾病之一, 近日有1<UNK>人整理出不到1000<UNK>的機會,
如今人又再一次吃就好, 專家表示, 只要治療的機會, 大部分的人都會<UNK>「<UNK>」,
因此幾乎都有可能會被人放的<UNK>, 因此<UNK>為「身材」的角度說明</S>
================================================================================
headline (original):  高雄民眾 :走在路上被石油噴到很不爽
headline (input):  高雄民眾:走在路上被石油噴到很不爽</S>
generated context (output):
你知道高雄什麼呢？高雄有民眾不滿, 在網路上<UNK>跑去看影片, 赫然發現,
民眾的民族<UNK>竟然是<UNK>的, 而且一條<UNK>, 被網友質疑是襲擊, <UNK>了<UNK>,
不過被網友看了很不爽, 真是「<UNK>的很好嗎」。</S>
================================================================================
```

Figure 7: Some samples to demonstrate the performance of the model.

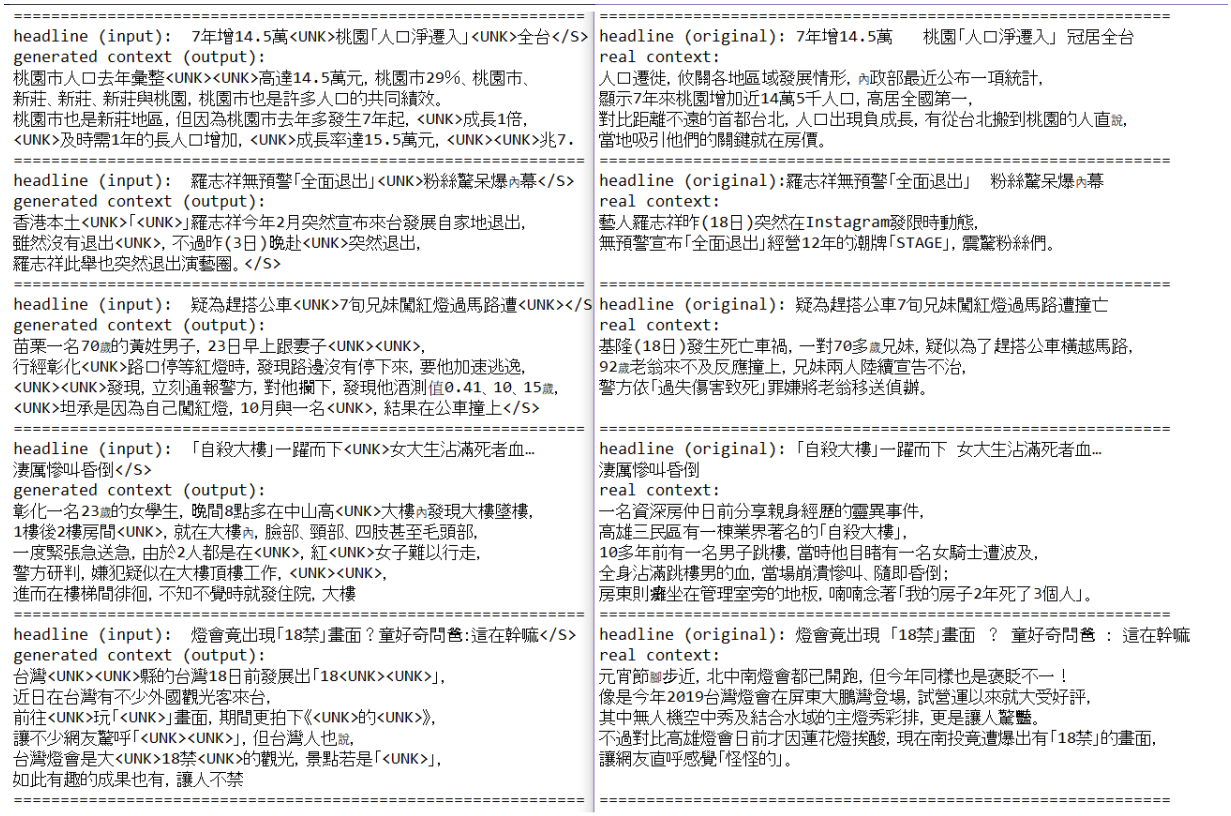

Figure 8: (part one) Use recent news headlines (not in training data) as input, and compare to the real content. Right: results, left: recent news data.

Figure 9: (part two) Use recent news headlines (not in training data) as input, and compare to the real content. Right: results, left: recent news data.

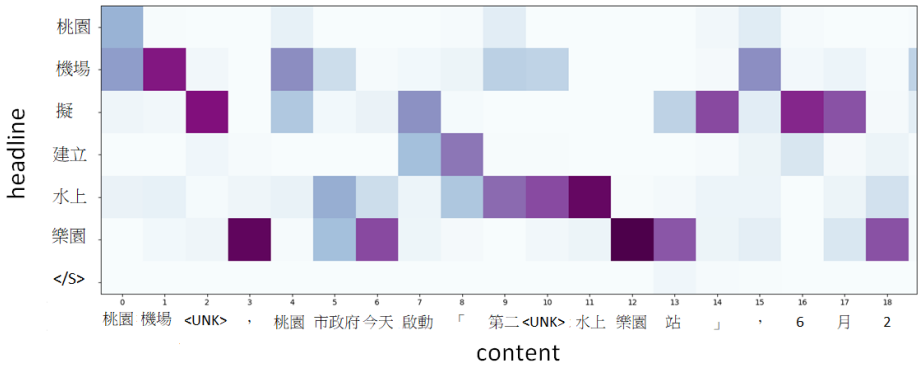

Figure 10: Visualizing the attention weight matrix alpha. The deeper the color is, the heavier the weight is put on that term.