Generating News Headlines with Recurrent Neural Networks

ChungTing Tsai [1] **WenBin Han** [1]

1. Abstract

According to (Lopyrev, 2015), we use combine a recurrent neural network with different attention mechanisms to generate news headlines from the content of news articles. Recurrent neural network is an application of encoder-decoder with GRU units. Given different attention mechanism, we evaluate the results of news headlines. We modeled after (Lopyrev, 2015) and tried to improved the system. The results show that the model is quite effective given concisely paraphrasing news articles. In particular, the combination of copying attention and MLP attention performs better.

2. Background

For many tasks, recurrent neural networks is effective, especially sequence data, such as machine translation and speech recognition. These models are trained on large amounts of input and expected output sequences, and then are able to generate output sequences with some rare words or words never seen before in the training data. Recurrent neural networks have also been applied recently to reading comprehension. Therefore, the models are trained to recall facts or statements from input text. Our work is closely related to whoever also use a neural network to generate news headlines using the same dataset as this work.

3. Introduction

Summarization is the task of abstracting a piece of article to a shorter version that contains the main information from the original. There are two broad approaches to summarization: extractive and abstractive. Extractive methods assemble summaries from passages taken directly from the source text, while abstractive methods may generate novel words and phrases not seen in the source text as a human-written abstract usually does. The extractive approach is easier because copying large chunks of text from the source document ensures baseline levels of grammaticality and accuracy. On the other hand, sophisticated abilities that are crucial to high-quality summarization, such as paraphrasing, generalization, or the incorporation of real-world knowledge, are possible only in an abstractive framework.

In this paper, we use a general RNN architecture with different copying mechanism from (Abigail Seev, 2017). Besides, we also fed the model on pre-train word embeddings to improve. Our project is separated into three subtasks. The first subtask determined whether copying attention mechanism works or not. The second subtask focused on whether feeding pre-train word embeddings affects or not. The last subtask compared the different formulas used in attention mechanism.

4. Model

We follow the encoder-decoder architecture described in (Lopyrev, 2015). The architecture consists of two parts - an encoder and a decoder - both recurrent neural networks. In each, we use GRU units and attention mechanism. In encoder, we use bidirectional recurrent neural network, by which the model increase the amount of input information available to the network. The principle of bidirectional RNN is to split the neurons of a regular RNN into two directions, one for positive time direction(forward states), and another for negative time direction(backward states). By using two time directions, input information from the past and future of the current time frame can be used unlike standard RNN which requires the delays for including future information.

5. Attention Mechanism

An attentional mechanism has lately been used to improve neural machine translation (NMT) by selectively focusing on parts of the source sentence during translation. In (Minh-Thang Luong, 2015), they proposed two simple and effective classes of attentional mechanism: a global approach which always attends to all source words and a local one that only looks at a subset of source words at a time. By attention mechanism, we can concentrate on some important words rather than pay equal attention to each input word.

[*]Equal contribution [1]National Tsing Hua University, Hsinchu, Taiwan. Correspondence to: ChungTing Tsai <ting@nlplab.cc>, WenBin Han <vincent.han@nlplab.cc>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

5.1. Global attention

The idea of a global attentional model is to consider all the hidden states of the encoder when deriving the context vector c_t. In this model type, a variable-length alignment vector a_t, whose size equals the number of time steps on the source side, is derived by comparing the current target hidden state ht with each source hidden state $\bar{h_s}$

$$a_t(s) = align(h_t, \bar{h_s}) = \frac{exp(score(h_t, \bar{h_s}))}{\sum_{s'} exp(score(h_t, \bar{h_{s'}}))}$$

Here, score is referred as a content-based function for which we consider three different alternatives:

$$score(h_t, \bar{h_s}) = \begin{cases} x = h_t^\top & dot \\ y = h_t^\top W_a \bar{h_s} & general \\ z = v_a^\top tanh(W_a[h_t' \bar{h_s}]) & concat \end{cases} \quad (1)$$

5.2. Copying attention

Sometimes our model faces OOV(out of vocabulary) problem, which means we get '<UNK>' in decoding. However, through copying attention mechanism from (Jiatao Gu, 2016), the probability could be decreased.

Copying attention selective replicate some part of the input sequence in the output sequence. From a cognitive perspective, the copying mechanism is related to rote memorization, requiring less understanding but ensuring high literal fidelity. From a modeling perspective, the copying operations are more rigid and symbolic, making it more difficult than soft attention mechanism to integrate into a fully differentiable neural model.

Therefore, the probability of generating any target word is given by the mixture of probabilities, generated mode or copy mode.

6. Data

6.1. Dataset link

Training and evaluation data is available on summary.tar.gz. The compressed file contains four directory: DUC2003, DUC2004, Giga, and train. Train directory consists of training dataset and validation dataset. In DUC2003, DUC2004, and Giga file folder, each line in input.txt is the first paragraph of its article, and each line in task_ref.txt is corresponding title. Giga dataset is used as test data.

6.2. Usage

Under the project directory, command 'python3 script/train.py' can run the program and reproduce the experiments shown below. After the command, it will start to read training data assigned before. After 300,000 steps, it will terminate automatically. Besides, the program will save the model every 20,000 steps.

Basically, it will run test.py automatically. However, if you terminate it during the training process, you can execute 'python3 script/test.py' directly. It will run the testing process with the most up-to-date model given the test data. Moreover, you can select different test data, such as giga, duc2003, or duc2004, to generate their titles with beam_size in 1 and 10.

7. Evaluation

In our experiment, we have three comparisons. First, we focused on whether copying attention mechanism helps or not. If so, the second experiment would use copying attention with general attention to compare if feeding pre-train word embeddings improves the model or not. Last, we did some research on different alternatives - dot, general, and MLP, of calculating attention.

7.1. Experiment on Copying Attention

To examine the effect of copying attention, we use general attention mechanism as our baseline, and the comparison is general attention mechanism with copying attention. For our model settings, the size of word embeddings is 256. Encoder and decoder use 2 hidden layers of LSTM with 512 hidden units for each layer. Besides, encoder is bidirectional RNN and decoder is general RNN. In addition, we set dropout 0.3 and use Adam optimizer with learning rate of 0.01.

After an epoch, the training process produced a model, whose name contains the accuracy and perplexity of validation dataset. As Figure 1 shown, the accuracy increases with the number of epoch but it converges after 12 epochs. However, for evaluation, we still used the model generated on epoch 16.

The first table in Figure 4 shows the results of evaluation. The model is improved a little by copying attention mechanism. In consequence, we take advantage of it in each model.

7.2. Experiment on Pre-train Word Embeddings

After the first experiment, we add copying attention mechanism in the following research. We are curious about if giving pre-train word embedding helps or not. Therefore, we utilized pre-trained word vector by GloVe method and Wikipedia data. However, the size of pre-train word embedding is 100, different from original size 256. The rest of settings is the same as previous experiment. Encoder and decoder use 2 hidden layers of LSTM with 512 hidden units for each layer. Besides, encoder is bidirectional RNN and decoder is general RNN. In addition, we set dropout 0.3 and use Adam optimizer with learning rate of 0.01.

After each epoch, the training process released a model, whose name contains the accuracy and perplexity of validation dataset. As Figure 2 shown, the accuracy still increases with the number of epoch but it almost converges after 11 epochs. However, for evaluation, we still used the model generated on epoch 16 to predict.

Second table in Figure 4 shows the results of evaluation. The model is improved a little with pre-trained word embedding. Consequently, will applied pre-trained word embedding to every model. =====

7.3. Experiment on alternatives of calculating attention score

Based on the previous result, all the models use in this experiment. Moreover,

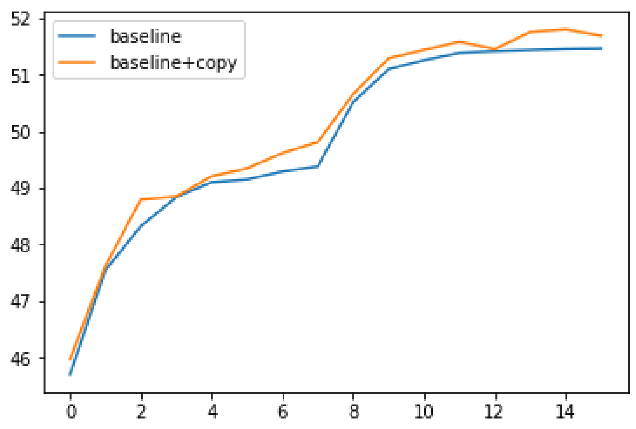
We are curious about if giving pre-train word embedding helps or not. Therefore, we utilized pre-trained word vector by GloVe method and Wikipedia data. However, the size of pre-train word embedding is 100, different from original size 256. The rest of settings is the same as previous experiment. Encoder and decoder use 2 hidden layers of LSTM with 512 hidden units for each layer. Besides, encoder is bidirectional RNN and decoder is general RNN. In addition, we set dropout 0.3 and use Adam optimizer with learning rate of 0.01.

After each epoch, the training process released a model, whose name contains the accuracy and perplexity of validation dataset. As Figure 3 shown, the accuracy still increases with the number of epoch but it almost converges after 11 epochs. However, for evaluation, we still used the model generated on epoch 16 to predict.

Second table in Figure 4 shows the results of evaluation. The model is improved a little with pre-trained word embedding.

7.4. Goal

The target of the project is to summarize the news article and generate its suitable title. Researcher usually use whole articles as input; however, in this project, we only remain the first paragraph as input.



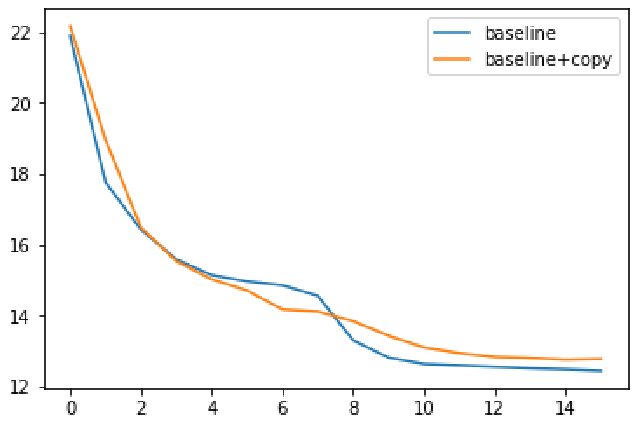


Figure 1. The accuracy and perplexity training model

7.5. Network Structure

7.5.1. Data preparation(data_util.py)

We produced two dictionaries, doc_dict.txt and sum_dict.txt, which contain 80000 high-frequency words in "data/train.article.txt" and "data/train.title.txt" individually. Both dictionaries are in descending order according to the count of the words. Besides, the dictionaries also include four specific IDs, which are ID_PAD, ID_UNK, ID_EOS, and ID_GO.

First, we pad the input with ID_PAD to match the closest bucket size, which will be described in detail later. ID_GO and ID_EOS are used to notate the begin and end of sentences. Those words do not appear in the dictionaries will be replaced with ID_UNK.

7.5.2. Bucketing

Although TensorFlow r.1.0 provides dynamic RNN seq2seq framework, which is much easier to understand than the tricky bucketing mechanism in order to avoid wasting computation. In this project, we use dynamic RNN to generate computing graph, and we still split the dataset into

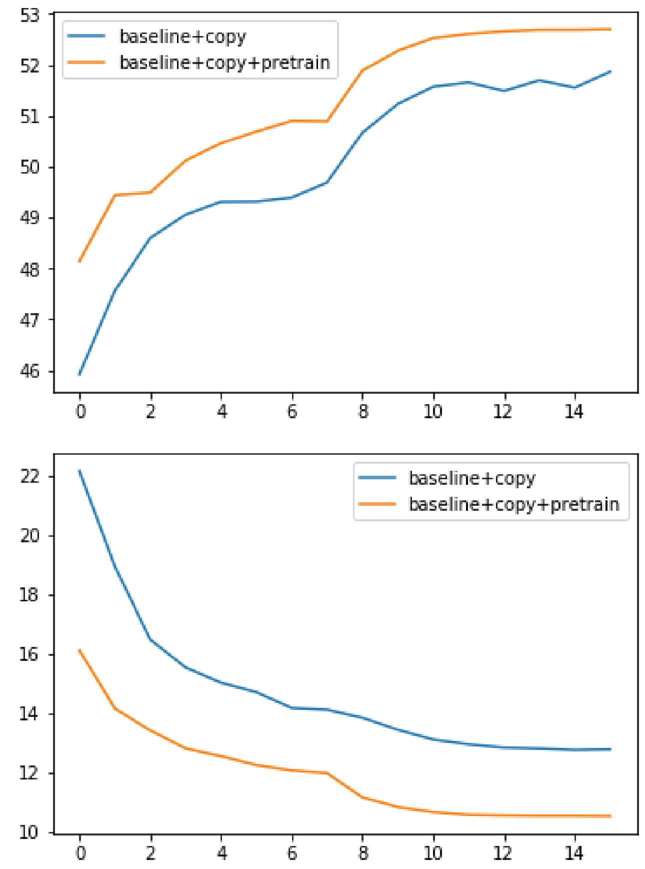


Figure 2. The accuracy and perplexity training model

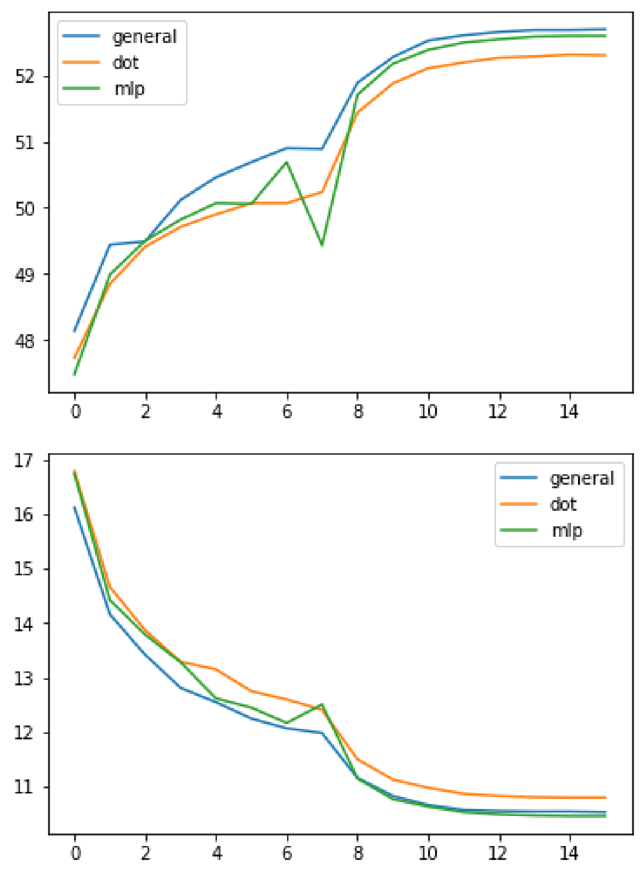


Figure 3. The accuracy and perplexity training model

several buckets. For those insufficient length of sentences, we use ID_PAD to fill up the shortage. Last, each bucket is one batch as input.

7.5.3. Embedding

We use 200 dimensional word2vec trained on the same corpus during training to initialize the model embedding.

7.5.4. Encoder and Decoder

This project is an implementation of sequence-to-sequence model using a bidirectional GRU encoder and a GRU decoder.

In encoder, because of different lengths of sentences, we tell dynamic_rnn the exact lengths of source sentences via 'source_sequence_length' parameter, such as source_sequence_length = batch_size.

In addition, we replace unidirectional dynamic rnn with bidirectional dynamic rnn. By doing so, we can get the forward and backward hidden information in the same time.

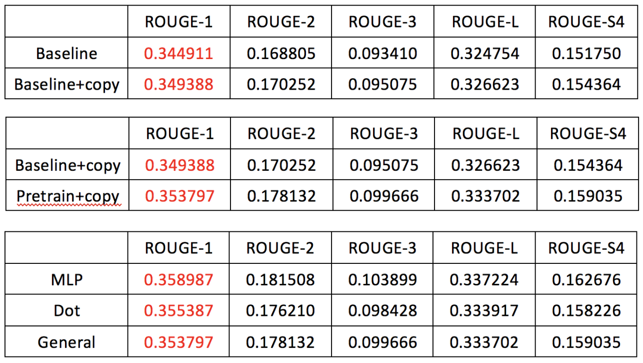
On the other hand, in decoder, time and batch dimensions are dynamic. They can change from batch to batch in runtime state. Therefore, how far the decoder can run depends on how we use ID_GO and ID_EOS.

7.5.5. Attention Mechanism

The attention mechanism follows Bahdanau et. al. We implement it in tf.contrib.seq2seq. Moreover, we refine the softmax function in attention so that those paddings will always get 0 score, which means paddings will not be chosen as predicted word.

7.5.6. Beam Search

For simplicity and flexibility, beam search algorithm is completed in python while the network part is left in tensorflow. In testing process, we consider batch_size as beam_size. Because tensorflow graph will generate only 1 word, program will create a new batch according to the result. By iteratively doing so, beam search result is generated.



	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-L	ROUGE-S4
Baseline	0.344911	0.168805	0.093410	0.324754	0.151750
Baseline+copy	0.349388	0.170252	0.095075	0.326623	0.154364

	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-L	ROUGE-S4
Baseline+copy	0.349388	0.170252	0.095075	0.326623	0.154364
Pretrain+copy	0.353797	0.178132	0.099666	0.333702	0.159035

	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-L	ROUGE-S4
MLP	0.358987	0.181508	0.103899	0.337224	0.162676
Dot	0.355387	0.176210	0.098428	0.333917	0.158226
General	0.353797	0.178132	0.099666	0.333702	0.159035

Figure 4. The evaluation of the model with dataset Giga

8. Result

8.1. Table

In order to evaluate whether generated titles are good or not, we use a popular method, **BLEU** (bilingual evaluation understudy) score. BLEU is an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another. However, in this case, summarization is also one kind of translations. The main idea of BLEU is that ”the closer a machine translation is to a professional human translation, the better it is.”

The concept of BLEU concentrates on n-gram, and calculates score by precision, which is the probability of n-gram from candidate showing up in reference sentence. To be more specific, the more unique patterns in one predicted headline exist in true headline, the higher score it obtains.

As the table**??** shown, given several test datasets and two kinds of beam size, giga dataset can get the highest average score, 0.328, with beam size 1.

Moreover, in Table4, we picked up some predicted headline to do comparison. For several articles, the model can generate totally same headline as actual headline. Despite the fact that the average is not high enough, the model can retrieve the main information and produce the headline on some articles. However, in some cases, the BLEU scores are zero. There are still some problems in acquiring important information from some articles. Besides, after observing the data, we found that several sentences are weird, such as ”-lrb- graphic -rrb- UNK”, which is not a correct sentence. In general, there is still much room for improvement and we will try to apply different hyper-parameters to the model in the future.

8.2. Graph for loss/epoch

We use tensorboard to visualize the loss trending. Picture 5 is the loss plot with 0.6 smoothing. In the picture, we can obtain that the model improved in 50,000 steps, and became stable after 100,000 steps. Therefore, to accelerate the training process, we will set max steps 300,000 in the future.

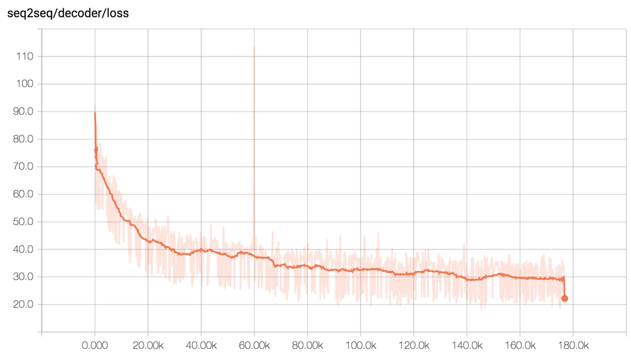


Figure 5. Loss in the range 350,000 steps with smoothing 0.6. The model became stable after 150k steps.

8.3. Screen Shot for program

The first part of program we intend to explain is function create_bucket, shown in picture**??**. Every bucket contains upper-bound length of sentence and headline, which will be in pair format. There are five buckets: (30, 10), (50, 20), (70, 20), (100, 20), (200, 30). After we transformed every word into index stored in array, we pass them into create_bucket function. It will calculate the length of each sentence and headline and store in correct bucket via if-condition. Afterwards, we pick one of the bucket randomly, and get one batch from that bucket. Picture7 shows the loss information during training process. In each step, we can observe the average loss on training data and the losses for each bucket on validation data.

Second, for building encoder, we use built-in function ’bidirectional_dynamic_rnn’ in tensorflow with GRU cell. If computational resource is enough, you can choose basic LSTM cell when creating model by passing ’use_lstm=True’. In addition to encoder, instead of bidirectional dynamic rnn, we only use dynamic rnn as our decoder model because decoder does not focus on back-propagation.

Table 1. The ROUGE evaluation of the model given Giga

Giga	baseline	MLP-Pre	Dot-Pre	General-Pre
ROUGE-1	0.344911	0.358987	0.355387	0.351981
ROUGE-2	0.168805	0.181508	0.176210	0.175709
ROUGE-3	0.093410	0.103899	0.098428	0.099610
ROUGE-L	0.324754	0.337224	0.333917	0.330616
ROUGE-S4	0.151750	0.162676	0.158226	0.157962

Table 2. The ROUGE evaluation of the model given DUC2004

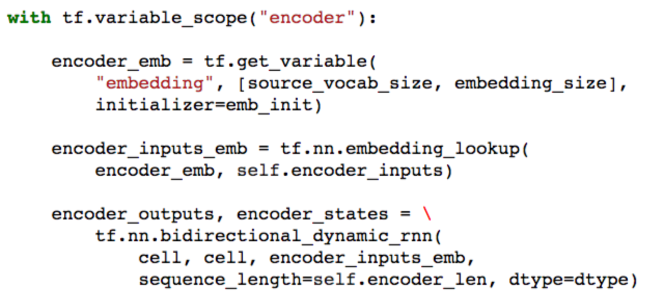
DUC2004	baseline	MLP-Pre	Dot-Pre	General + Pre
ROUGE-1	0.282147	0.289550	0.297677	0.292907
ROUGE-2	0.102883	0.107201	0.109321	0.106579
ROUGE-3	0.037308	0.043926	0.042219	0.040940
ROUGE-L	0.256304	0.263968	0.268769	0.266695
ROUGE-S4	0.082818	0.089125	0.090068	0.088935

Table 3. The ROUGE evaluation of the model given DUC2003

DUC2003	baseline	MLP-Pre	Dot-Pre	General + Pre
ROUGE-1	0.274406	0.277940	0.284574	0.282747
ROUGE-2	0.098237	0.098995	0.104592	0.101964
ROUGE-3	0.037699	0.040994	0.043402	0.043293
ROUGE-L	0.253070	0.258083	0.263891	0.261798
ROUGE-S4	0.080960	0.080806	0.085617	0.084001

score	predicted headline	true headline	context
0.597	thai army chief tells protesters to leave key sites	thai protesters must leave airport other sites : army chief	”thailand ’s powerful army chief wednesday told anti-government protesters they must leaveseveral key sites that they are occupying , including bangkok ’s international airport .”
0.6	chinese defector arrives in china	chinese plane arrives in philippines	”an unscheduled flight from xiamen in china which could be carrying north korean defector hwang jang-yop landed tuesday at the clark airbase , north of here , sources at the manila international airport said .”
0.751	tigers admit losing ## in suicide attack	tigers admit losing ## in suicide attack on navy	tamil tiger guerrillas monday admitted losing ## men in a weekend suicide attack against the navy in northern sri lanka which may have set back military plans for a fresh offensive against them .
0.846	israel prepares for rabin state funeral	israel prepares jerusalem state funeral for rabin	israel prepared sunday for prime minister yitzhak rabin ’s state funeral which will be attended by a host of world leaders , including us president bill clinton and the jordanian and egyptian heads of state .
0.857	skorea posts current account surplus in october	skorea posts current account surplus for october	south korea posted a current account surplus of #.# billion dollars in october , the central bank said thursday , a figure that is likely to relieve pressure on the shaky won .
1.0	jakarta shares close #.# percent lower	jakarta shares close #.# percent lower	jakarta share prices closed #.# percent lower tuesday amid selling pressure on heavyweight stocks , brokers said .

Table 4. Comparison between predicted headline and true headline



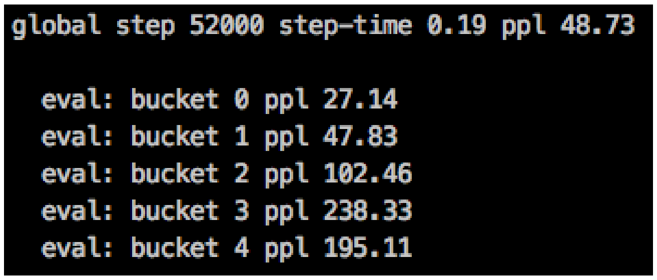
```
with tf.variable_scope("encoder"):

    encoder_emb = tf.get_variable(
        "embedding", [source_vocab_size, embedding_size],
        initializer=emb_init)

    encoder_inputs_emb = tf.nn.embedding_lookup(
        encoder_emb, self.encoder_inputs)

    encoder_outputs, encoder_states = \
        tf.nn.bidirectional_dynamic_rnn(
            cell, cell, encoder_inputs_emb,
            sequence_length=self.encoder_len, dtype=dtype)
```

Figure 6. Encoder model is using bidirectional_dynamic_rnn in tensorflow. Before creating model, we will transform words into vector using built-in function 'embedding_lookup' in tensorflow.



```
global step 52000 step-time 0.19 ppl 48.73

  eval: bucket 0 ppl 27.14
  eval: bucket 1 ppl 47.83
  eval: bucket 2 ppl 102.46
  eval: bucket 3 ppl 238.33
  eval: bucket 4 ppl 195.11
```

Figure 7. In each step, we can see average perplexity on training data and five buckets' perplexity on validation data.

References

Abigail Seev, Peter J. Liu, Christopher D. Manning. *Get To The Point: Summarization with Pointer-Generator Networks*. PhD thesis, Department of Computer Science, Stanford University, 2017.

Jiatao Gu, Zhengdong Lu, Hang Li Victor O.K. Li. *Incorporating Copying Mechanism in Sequence-to-Sequence Learning*. PhD thesis, Department of Electrical and Electronic Engineering, The University of Hong Kong, 2016.

Lopyrev, Konstantin. *Generating News Headlines with Recurrent Neural Networks*. PhD thesis, Department of Computer Science, Stanford University, 2015.

Minh-Thang Luong, Hieu Pham, Christopher D. Manning. *Effective Approaches to Attention-based Neural Machine Translation*. PhD thesis, Department of Computer Science, Stanford University, 2015.