Image Captioning Using LSTM Network

Nan Wei
UC San Diego
nwei@ucsd.edu

Renjie Shao
UC San Diego
reshao@ucsd.edu

Hongyi Ling
UC San Diego
holing@ucsd.edu

Abstract

In this article we are dealing with image captioning task on COCO dataset. Here we use a encoder-decoder model while pretrained Resnet50 acts as encoder and LSTM/RNN as decoder. We evaluate our generated captions by perplexity and BLEU score. Our best result reach 9.96 Perplexity, 87.33 BLEU-1 score and 19.02 BLEU-4 score, which verifies our model's performance.

1 Introduction

Image captioning is a task to generate captions from images. Normally we want a model to learn from a labeled dataset, i.e. images with captions, can caption new images automatically. To handle this problem, usually we need an encoder to extract features from images and then a decoder to generate texts fro extracted features. In this task we will use pretrained Resnet50 [1] as our encoder. For decoder, we will use LSTM/RNN to generate texts. The dataset we use is COCO [2].

2 Method

In this section we will talk about the architecture of our model. The framework of our model is shown as Figure 1. We feed our images to ResNet and get an output vector of 2048 dimensions. Then we use a linear layer to map our feature vector to 300 dimensions, which is the embedding dimensions. This 300 dimensions vector will act as input to feed into our LSTM/RNN network. Then we use teaching force to train our decoder, for each time step, we use current label word as next time input to our decoder, i.e. $x(t+1) = target(t)$. Note that each word first go through the embedding layer, which will transfer them into 300 dimensions vector.

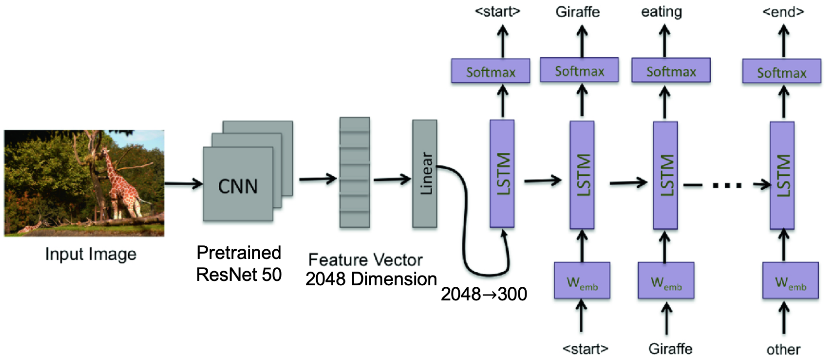

Figure 1: Model Framework [3]

In this task we will try LSTM and RNN as our decoder and compare the performances between them. Also, we will also use pretrained word embedding to see if this may improve our performance.

The word embedding we use here is GloVe.6B with 300 embedding dimensions [4]. For baseline LSTM and RNN model, we use random initialized word embedding and fine tuning during the training process. Pretrained word embedding are widely used in NLP tasks. The advantages to use pretrained word embeddings is that it usually trained in a vary larger dataset with longer training process, which enhances the expressive capacity of each word vector. For example, the GloVe.6B we use are trained on 6 billion tokens. Hopefully, we want to see improvement after using this word embedding.

The detailed parameters and hyper-parameters for each models are listed in Table 1. Since LSTM has more parameters than RNN at each unit, here we use 2 layers RNN to make a compromise.

Model	Encoder	Embed Dim	Hidden Dim	Layer	Optimizer	LR	L_2 Penalty
LSTM(baseline)	Resnet50	300	512	1	Adam	0.001	10^{-5}
RNN	Resnet50	300	512	2	Adam	0.001	10^{-5}
LSTM+GloVe	Resnet50	300	512	1	Adam	0.001	10^{-5}

Table 1: Hyper-parameters for different models

3 Experiments

In this section, we will show the training process for each models and compare the performances in different indices such as perplexity, BLEU score.

3.1 Training curve

Firstly let us see the training curve for baseline LSTM and RNN, which is shown as Figure 2, 3.

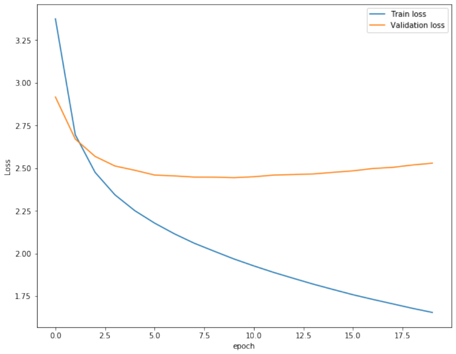

Figure 2: Training curve of LSTM

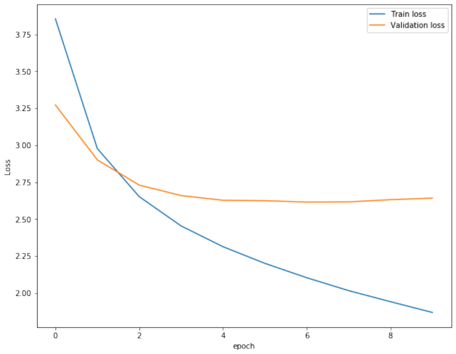

Figure 3: Training curve of vanilla RNN

As we can see, LSTM converges at 10 epochs while RNN at 6. Though our RNN has two layers and LSTM only has one, it still takes more time for LSTM to converge. However, the validation loss of LSTM is slightly better than RNN. Firstly, a LSTM cell unit has more parameters than RNN, so it is reasonable LSTM need more time to train. Also since LSTM has stronger ability to catch long-term dependency, it performs better than RNN in loss.

3.2 Pretrained Word Embedding

Here we will see the learning curve of LSTM with pretrained GloVe word embeddings. The learning curve is shown as Figure 4. As we can see, LSTM with pretrained word embedding converges faster than baseline LSTM. This is because by using pretrained word embeddings, we do not need much time to train our embedding layer. The validation loss of both model is near, which implies that by several epochs our baseline model can learn a good embedding for words.

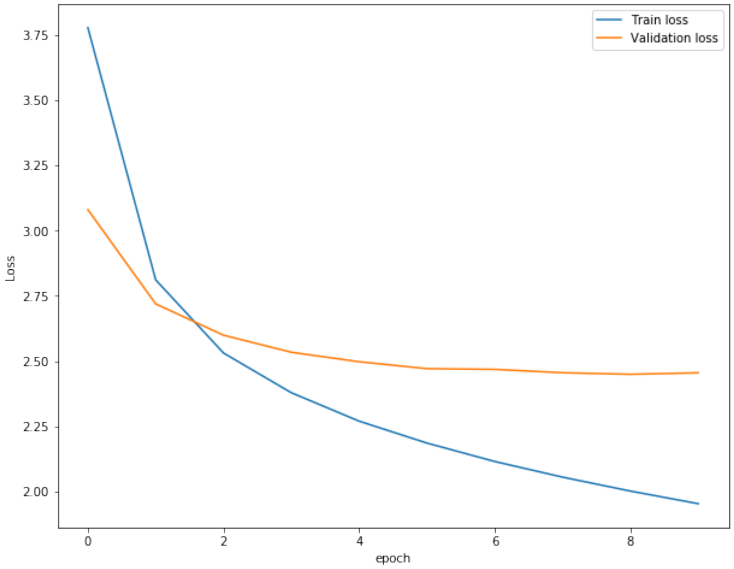

Figure 4: Training and validation loss of LSTM with pretrained embedding

3.3 Perplexity and BLEU

Now we move on to the evaluation of our models. In generation tasks, perplexity is a widely used index to evaluate the generated texts. Its definition is

$$ppl = \exp(-\sum_x p(x)\log p(x)). \tag{1}$$

Perplexity is a measurement of how well a probability distribution or probability model predicts a sample. The smaller perplexity we have, the better prediction it will give. So perplexity gives a probability evaluation of our model. BLEU (bilingual evaluation understudy) [5] is another way to measure our model. It use n-gram model to compute the correlation of machine output and labels. Here we use BLEU-1 and BLEU-4 scores to evaluate our models.

The results of different model is shown as Table 2. As we can see, LSTM models has better performance than RNN in all indices. It proves that, with the capacity of catch long-term dependency, LSTM is more powerful than RNN. Also note that LSTM-GloVe is slightly better than baseline model. It shows that with efficient training, our baseline can learning a good word embedding. And also note that in BLEU-4, LSTM-GloVe has better performance. So pretrained word embeddings still show improvement. It is a good idea to try pretrained word embedding when doing NLP tasks.

Model	Loss	Perplexity	BLEU-1	BLEU-4
LSTM	2.29	9.96	87.33	17.04
LSTM+GloVe	2.30	9.97	87.31	19.02
RNN	2.46	11.68	87.04	17.22

Table 2: Performance for different models

3.4 Stochastic Approach

When choosing the generated word, formerly we always choose the one with maximum probability. By this way we always generates texts most likely to training set probability distribution. However, it also has disadvantages. The key problem of this method is that it will cause our generated corpus losing variety. To solve this problem, here we will try a stochastic approach, i.e., to sample words with weights as

$$y^j = \exp\left(o^j/\tau\right) / \sum_n \exp\left(o^n/\tau\right), \tag{2}$$

where o^j is the output of last layer, n is the size of vocabulary and τ is the hyper-parameter 'temperature'.

Here we try several different temperature and calculate the BLEU score on our baseline model. The result is shown in Table 3. We can see that when $\tau = 0.1$ the model performs best. When temperature is very small, stochastic approach is close to deterministic approach. With τ grows, the BLEU scores decrease. And also, all of them are worse than deterministic approach. This is because deterministic approach always make decision that lead generated texts as close to references as possible, so when evaluated on BLEU scores, it performs better. However, stochastic approach can still give variety to our generated corpus, which is useful in some real tasks.

Temperature	0.1	0.2	0.7	1	1.5	2
BLEU-1	87.29	87.22	87.05	85.89	72.19	55.73
BLEU-4	17.3	17.13	15.41	13.16	5.66	3.48

Table 3: BLEU scores for baseline model with stochastic approach

4 Individual Contribution

Nan Wei

I implement BLEU evaluation, run and test RNN model. I also write related parts in report.

Renjie Shao

I implement data loader, model framework and help debug. I also write the report.

Hongyi Ling

I implement the generation part of our model and do experiments on baseline model. In addition, I write parts of the report.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[3] Anurag Tripathi, Siddharth Srivastava, and Ravi Kothari. Deep neural network based image captioning. In Anirban Mondal, Himanshu Gupta, Jaideep Srivastava, P. Krishna Reddy, and D.V.L.N. Somayajulu, editors, *Big Data Analytics*, pages 335–347, Cham, 2018. Springer International Publishing.

[4] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[5] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.