Zhao Lutong (1002872), Tang Xiaoyue (1002968), Wang Zijia (1002885)

DL Image Captioning Project

26th April 2020

1.OVERVIEW

In the report, we will explain how we Implemented an image captioning with a GUI using MS COCO dataset and what are the changes we have made based on the tutorial code.

2.GOALS

1. Modify the tutorial code with the aim to improve accuracy (BLEU score).
2. Build a GUI to show an image and the predicted caption.

3.USAGE

Using GUI

```
conda install kivy -c conda-forge
python gui.py
```

Test with terminal/cml

```
python sample.py --image='png/example.png'
```

4.PREPARE MODEL

Option 1. Using pre-trained model

If you do not want to train the model from scratch, you can use a pre-trained model. You can download our 3 pre-trained models here and the vocabulary file here. You should extract pretrained_model.zip to `./models/` and vocab.pkl to `./data/` using `unzip` command.

Option 2. Train from scratch

1. Clone the repositories

```
cd ../
git clone https://github.com/pdollar/coco.git
cd coco/PythonAPI/
Make
python setup.py build
python setup.py install
cd ../../
```

2. Download the dataset

```
pip install -r requirements.txt
chmod +x download.sh
./download.sh
```

3. Preprocessing

```
python build_vocab.py
python resize.py
```

4. Train the model

```
python train.py
```

5.MODIFICATION

We have tried three experiments as following:

1. Minimum Word Count

We have adjusted the threshold for minimum word count so that we can filter out the uncommon words like jargons and train a model that is more general. We have generated two vocab (accessible here), one used a minimum word count of 4 (model 1), and the other one is 10 (model 2). As shown in part 6 results, the accuracy and bleu score for model 1 is slightly better than model 2. Therefore, a threshold of 4 is maybe a more suitable choice.

2. GRU

We have adapted the tutorial code from an LSTM model to a GRU model, as in the previous homework it displayed a better result than LSTM as a more modern model, and we know that GRU has less trainable parameters, so it can save training time.

3. Beam Search

Instead of the greedy search used in tutorial code, we implemented beam search. The beam search algorithm selects multiple alternatives for an input sequence as each timestep based on conditional probability.

4. Decoder Dropout and BatchNorm

In experiment 2, we added a batch-norm layer after the linear layer and in experiment 3, we added a dropout layer with dropout rate 0.2 before the linear layer.

5. Snowball Stemming

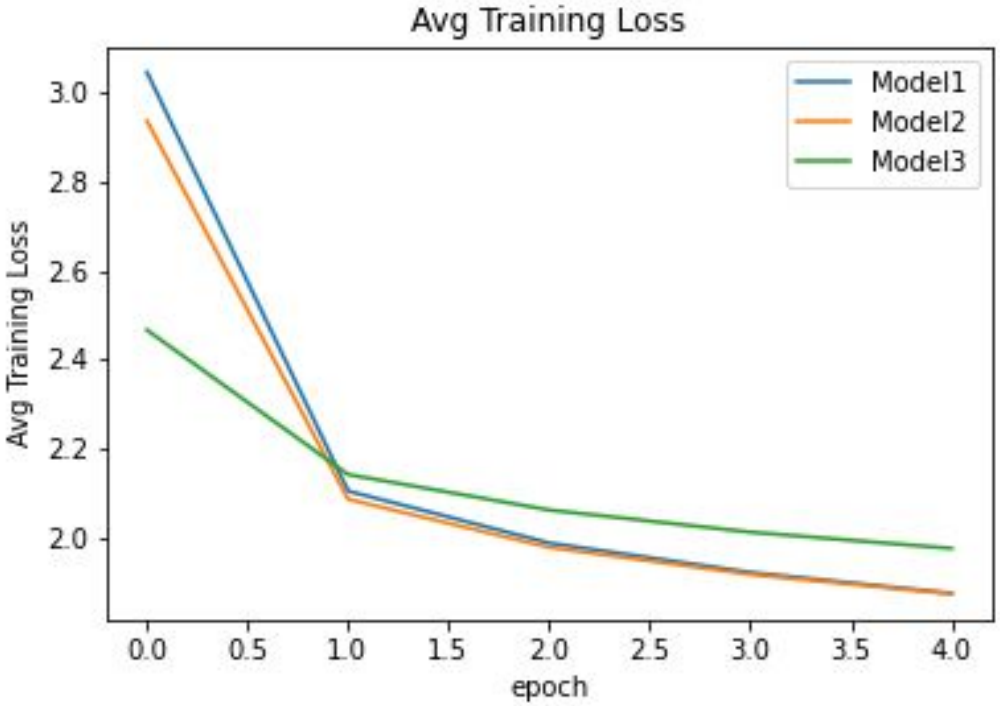
For all three experiments, we used Snowball stemmer to stem the word in the vocabulary, it is less aggressive than Lancaster stemmer while slightly improved from Porter stemmer. *However, this sometimes results in uncompleted words for prediction (e.g. computer, computing or computed to comput)*

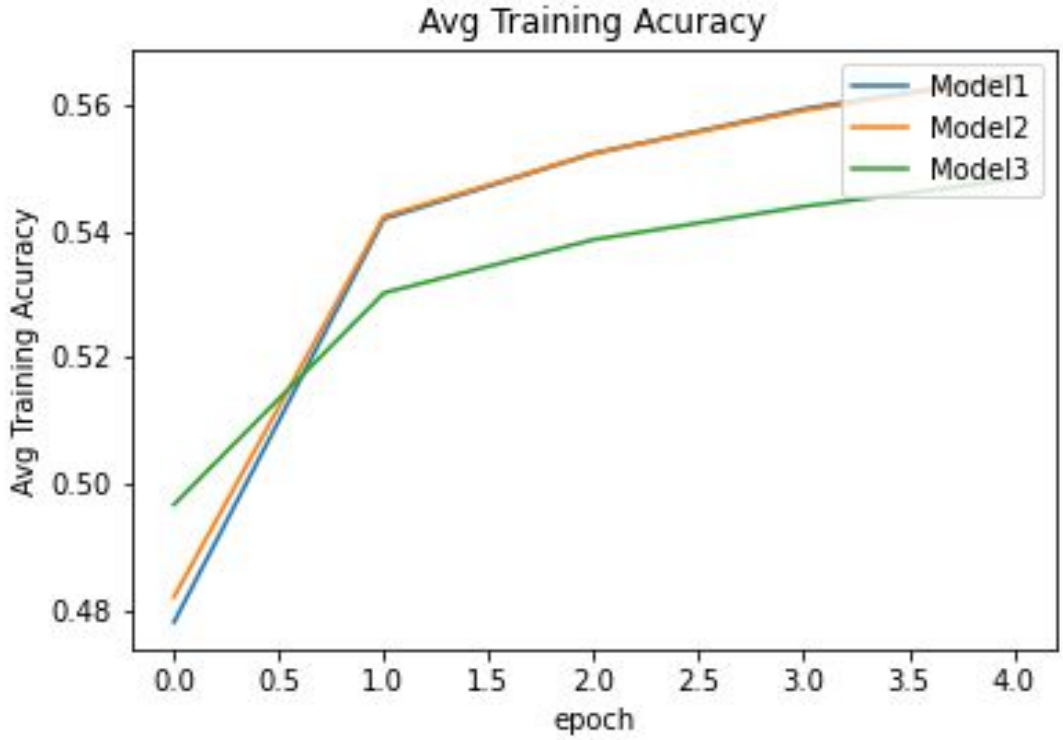
6. Encoder Model

For the encoder models, we experiment with ResNet50(model 1 and model 2) and ResNext_101_32*8d (model 3), as shown in part 6 results, the latter one delivers a slightly higher BLEU score than the ResNet50 model. We used a pre-trained weight that trained from ImageNet, and performed transfer learning on the encoder to improve the performance.

6.RESULTS

- Training

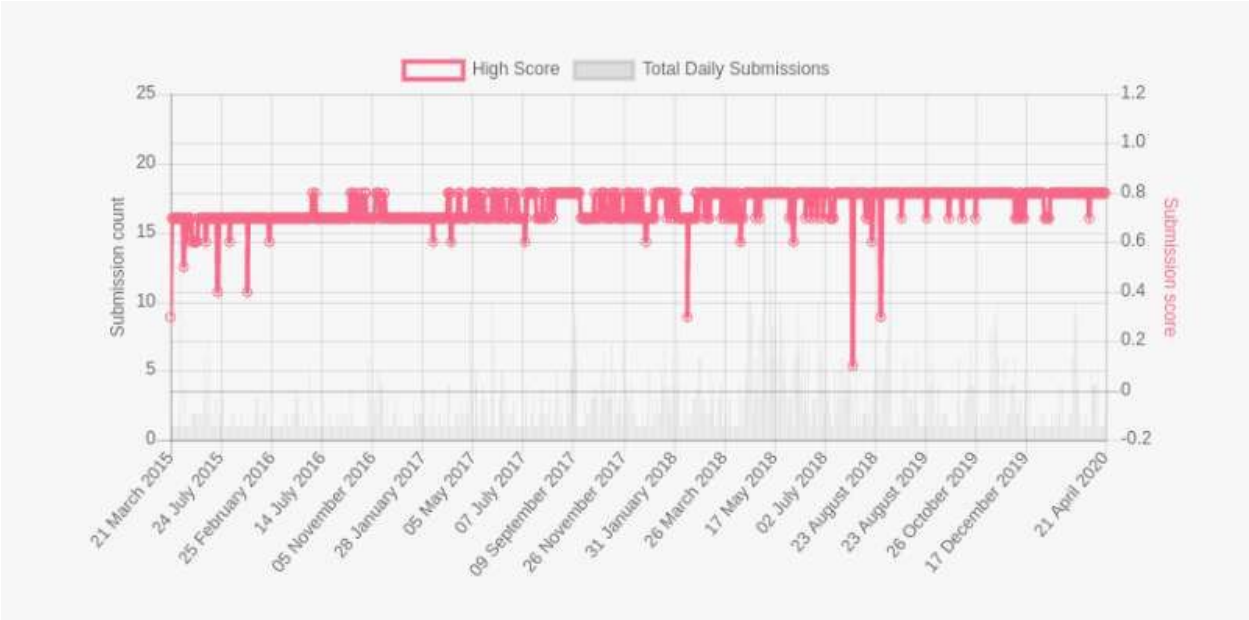

Avg Training Loss
Model1
Model2
Model3
Avg Training Loss
3.0
2.8
2.6
2.4
2.2
2.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
epoch


Avg Training Acuracy
Model1
Model2
Model3
Avg Training Acuracy
0.56
0.54
0.52
0.50
0.48
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
epoch

- Testing

Snowball Stemming & GRU	Beam Search	Decoder Dropout	Decoder BatchNorm	Minimum word count	Encoder Model	Training Accuracy	BLEU Score	Loss	Name
☑	☑	✖	☑	4	Resnet50	0.5647	0.570818/ 0.409930/ 0.291367/ 0.212085	1.8853	**Experiment 1**
☑	☑	✖	☑	10	Resnet50	0.5644	0.570178/ 0.409368/ 0.291380/ 0.212294	1.9067	**Experiment 2**
☑	☑	☑	✖	4	ResNext101_32x8d	0.5483	0.571144/ 0.410950/ 0.292646/ 0.21397	2.0077	**Experiment 3**

7.COMPARE AGAINST STATE-OF-THE-ART



Results

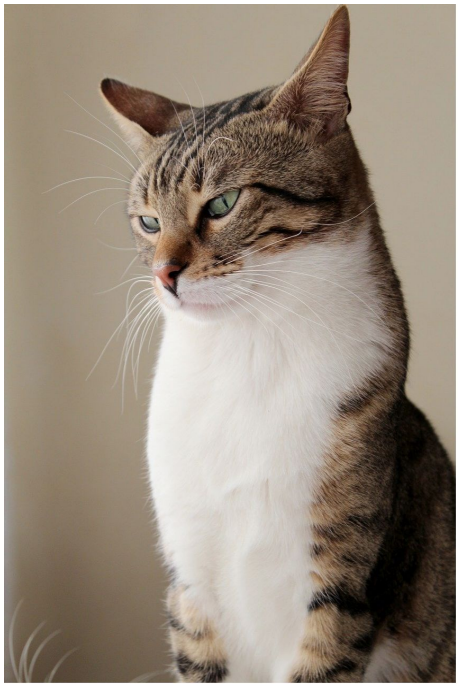
#	User	Entries	Date of Last Entry	BLEU-1		BLEU-2		BLEU-3		BLEU-4		METEOR		ROUGE-L		CIDEr-D	
				c5 ▲	c40 ▲	c5 ▲	c40 ▲	c5 ▲	c40 ▲	c5 ▲	c40 ▲	c5 ▲	c40 ▲	c5 ▲	c40 ▲	c5 ▲	c40 ▲
1	Yingwei.Pan	5	03/23/20	0.819 (2)	0.957 (4)	0.669 (2)	0.905 (3)	0.524 (3)	0.825 (3)	0.403 (3)	0.724 (4)	0.296 (1)	0.392 (1)	0.595 (3)	0.750 (2)	1.311 (1)	1.335 (1)
2	Meshed-Memory-Transformer	1	11/13/19	0.816 (5)	0.960 (1)	0.664 (5)	0.908 (1)	0.518 (5)	0.827 (1)	0.397 (5)	0.728 (1)	0.294 (4)	0.390 (2)	0.592 (5)	0.748 (5)	1.293 (4)	1.321 (2)
3	KingSoft_AILAB	1	10/24/19	0.819 (3)	0.957 (3)	0.670 (1)	0.906 (2)	0.527 (1)	0.825 (2)	0.406 (1)	0.725 (2)	0.294 (2)	0.388 (5)	0.598 (1)	0.750 (3)	1.298 (3)	1.317 (3)
4	luo3300612	1	04/14/20	0.813 (8)	0.958 (2)	0.660 (10)	0.905 (5)	0.515 (12)	0.822 (5)	0.393 (15)	0.723 (5)	0.293 (6)	0.388 (6)	0.589 (14)	0.742 (14)	1.290 (5)	1.314 (4)
5	IVA-HUAWEI	1	07/29/19	0.816 (6)	0.956 (8)	0.666 (4)	0.903 (6)	0.521 (4)	0.822 (6)	0.401 (4)	0.722 (6)	0.293 (7)	0.389 (4)	0.594 (4)	0.749 (4)	1.290 (6)	1.314 (5)

Image source

As shown above, our best model's BLEU-1 score (cumulative 1-gram) is 0.5711, which is quite lower than the state of art models, we think the reason could be: Firstly, our model is only trained for 5 epochs, it could be further improved if we train for more epochs. Besides, after reading the structure of the top-scoring model, we noticed many of them are using attention, and some of them are using ELMO for embedding. We think that's the main reason for their good performance.

8.FUNNIEST WRONG PREDICTION

<start> a cat is sit on a blue and white cat . <end>



<start> a bird fli over a lush green field . <end>

A successful example ! 👇

<start> a bird is perch on a branch of a tree . <end>

9.GROUP CONTRIBUTION

Group Member	Contribution
Wang Zijia	Experiment 1
Tang Xiaoyue	Experiment 2
Zhao Lutong	Experiment 3

APPENDIX - ORIGINAL PROCESS

Training phase

For the encoder part, the pre-trained CNN extracts the feature vector from a given input image. The feature vector is linearly transformed to have the same dimension as the input dimension of the GRU network. For the decoder part, source and target texts are pre-defined. For example, if the image description is "Giraffes standing next to each other", the source sequence is a list containing ['<start>', 'Giraffes', 'standing', 'next', 'to', 'each', 'other'] and the target sequence is a list containing ['Giraffes', 'standing', 'next', 'to', 'each', 'other', '<end>']. Using these source and target sequences and the feature vector, the GRU decoder is trained as a language model conditioned on the feature vector.

Test phase

In the test phase, the encoder part is almost same as the training phase. The only difference is that batch norm layer uses moving average and variance instead of mini-batch statistics. This can be easily implemented using encoder.eval(). For the decoder part, there is a significant difference between the training phase and the test phase. In the test phase, the GRU decoder can't see the image description. To deal with this problem, the GRU decoder feeds back the previously generated word to the next input. This can be implemented using a for-loop.