National Taiwan University Applied Deep Learning
Assignment 4
Machine Translation & Natural Language Generation
Chia-Hsuan-Li, Electrical Engineering

● Model Architecture

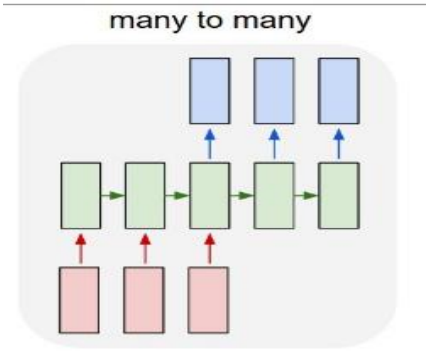
Embedding Layer：

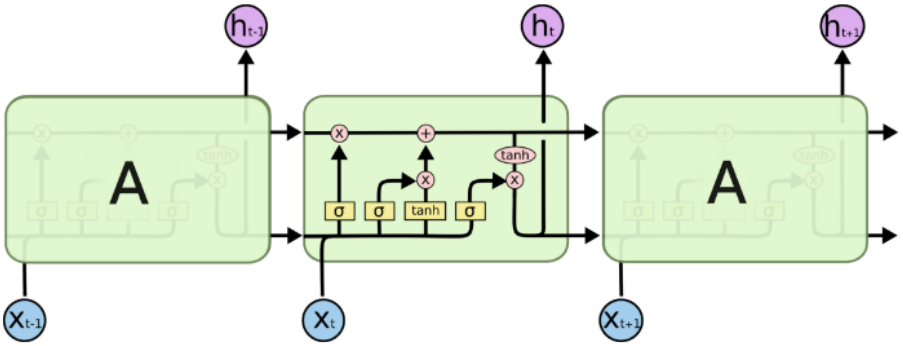
The embedding matrix will be initialized randomly and the model will learn to differentiate the meaning of words just by looking at the data.

Main Block：

The core of the model consists of an LSTM cell. The memory state of the network is initialized with a vector of zeros and gets updated after reading each word. And since we want our model to handle complicated task, we stack two layers LSTM to create a hierarchical feature representation of the input data.

The two tasks: Machine Translation and Natural Language Generation are both non-synced sequence to sequence problem. And the model described above is suitable to solve this task.





The repeating module in an LSTM contains four interacting layers.

Loss function：To avoid expensive computation cost in handling large vocabulary, we replace standard softmax loss with sampled softmax.

Bucketing and padding：To handle sentences of different lengths, we only use certain length and pad the short sentences with PAD symbol.
Evaluation：

The typical measure reported in the papers is average per-word perplexity (often just called perplexity), which is equal to

$$e^{-\frac{1}{N}\sum_{i=1}^{N}\ln p_{\text{target}_i}} = e^{\text{loss}}$$

and we will monitor its value throughout the training process.

If one could report a model perplexity of $2^{7.95}$ = 247 *per word.* In other words, the model is as confused on test data as if it had to choose uniformly and independently among 247 possibilities for each word.

● **Code description**
data_utils.py:
Creates the word-to-id vocabulary for both training data and dev data. And transform the whole document into ids.
seq2seq_model.py:
Where the model initialized. Create the internal multi-layer cell for our LSTMs. Implement the embedding layer for input words. Implement the attention mechanism.
translate.py:
Utilize the functions import from data_utils.py and seq2seq_model.py. Implement the bucket structure.

● **Experiment & Hyper-parameter Tuning**
Although these two tasks are trained on the same model, the hyper-parameter and corresponding performance are quite different.
Translation Task

MODEL	BLEU_SCORE
size64_layer2	**0.3**
size256_layer2	**0.36**
size256_layer3	**0.28**
size350_layer2	**0.41**

In translation task, basically the performance gets better as the LSTM size gets larger. My explanation for this is that the task is very complicated. The translation between two languages is hard and bear lots of varieties. And since the training data amount is enough, we can train on more complex model to get better performance.
Notice that layer3 is not better than layer2, this is probably because the gradient vanishes through multiple layers.
Basically the model takes more steps to be confident on predicting the loner sequence.
Below is a snapshot from the model size350_layer2.

```
global step 9000 learning rate 0.5000 step-time 0.32 perplexity 7.19
  eval: bucket 0 perplexity 2.92
  eval: bucket 1 perplexity 3.89
  eval: bucket 2 perplexity 7.39
  eval: bucket 3 perplexity 8.97
```

We stop training the model when all four buckets eval perplexity is single digit. While size256_layer3's bucket3 never reaches there.

Language Generation

MODEL	BLEU
SIZE128_LAYER2_version2	0.49
SIZE256_LAYER2_version2	0.53
SIZE400_LAYER2_learn0.3_version2	0.48
size256_layer2_learn0.5_version1	0.43
size300_layer_2_learn0.3_version1	0.47
size300_layer2_learn0.5_version1	0.43
ize256_layer2_ simplified	0.60
size300_layer2_ simplified	0.62

TRAINING DATA EXAMPLE FORMAT

Simplified

"inform(food=japanese,type=restaurant,pricerange=moderate)"

"i would like a food type pricerange priced"

version_2

inform name trattoria contadina pricerange moderate

trattoria contadina is a nice restaurant in the moderate price range

version_1 (AND REPLACE THE SPECIAL TOKENS WITH REAL INFO)

inform __name__ __pricerange__

__name__ is a nice restaurant in the __pricerange__ price range

Notice that all the BLEU score is evaluated by the simplified answer format. As for version1 output, the model predicts pretty good result by human observation. I strongly infer that the real BLEU for version1 output will be much higher than 0.62!
We can see how the LSTM size influence BLEU. 300 seems to be the peak and then decay either size larger or smaller.

- Reference

https://web.stanford.edu/class/cs124/lec/languagemodeling.pdf
Sequence-to-Sequence Models
https://www.tensorflow.org/versions/r0.12/tutorials/seq2seq/index.html
The Unreasonable Effectiveness of Recurrent Neural Networks
http://karpathy.github.io/2015/05/21/rnn-effectiveness/