# HW2-2 Chat BOT

## Model description

採用sequence-to-secuence的方式，input為一句話，output另一句話。

preprocessing:
根據training data出現的「字」建一字典，min count=100，建出來大小為3152(含<bos> <eos> <pad> <unk>)，接著把每一句話都轉成token後padding到一樣的長度(20)。

Embedding dim=256

Encoder:
1-layer uni-directional GRU，hidden units=256

Decoder:
1-layer uni-directional GRU，hidden units=256，沒有attenetion，initial state設為encoder的final state。
再把GRU輸出經過一層fully-connected layer變成3152維(字典大小)。

Loss:
對每一個字算cross entropy後「相加」成「每一句」的loss，再對整個batch做平均。

## How to improve your performance

這題採用在inference時做beam search的方式，希望會比用greedy效果來的好。
但是實際使用時發現，如果加上beam search，效果會變得很差，常常輸出同一句話。(結果在下一段，雖然分數是差不多的)
推設可能是因為model只有去學到正確的文法，但是忽略了前一句的情境(語意)，或許加上attention可以解決。

## Experimental results and settings

learning rate皆設為1e-3，沒有decay。batch size=256。epoch=50。Beam width=10。
這部份做了4種結果：(Optimizer={Adam, SGD}) X (Decoder={Greedy, Beam})

| perplexity | Greedy | Beam |
|---|---|---|
| SGD | 10.6 | 8.25 |
| Adam | 9.39 | 9.14 |

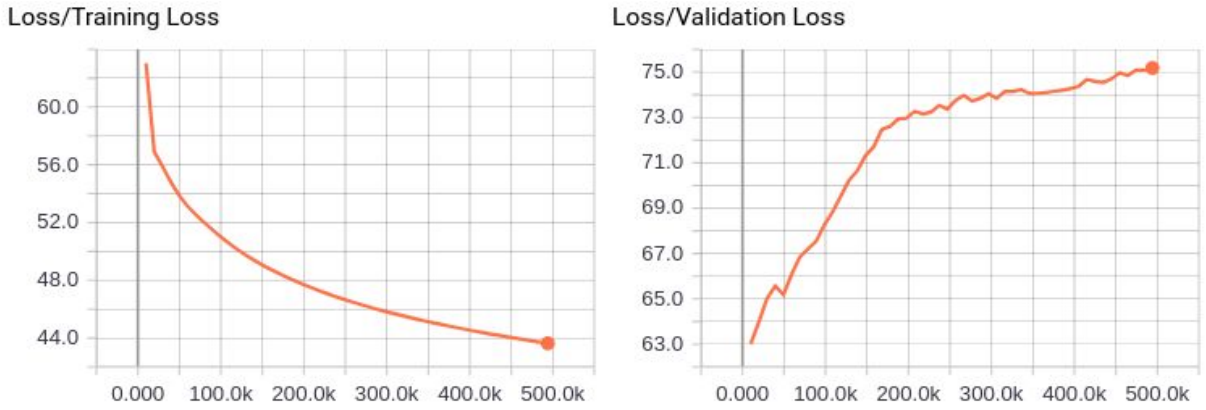
| correlation | Greedy | Beam |
|---|---|---|
| SGD | 0.31 | 0.3 |

| | | |
|---|---|---|
| Adam | 0.58 | 0.57 |

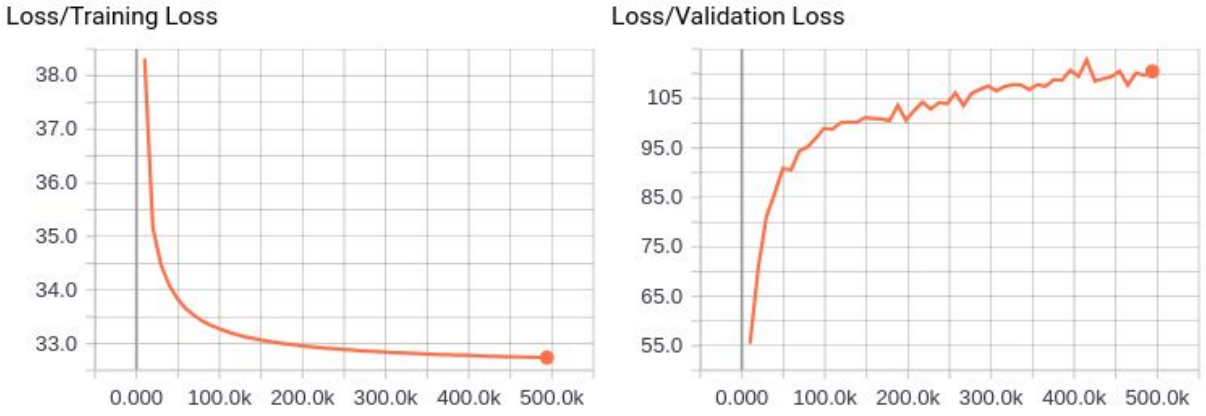
可以看到使用不同的Optimize方法對於結果的影響有顯著的差異

Training Curve

SGD:




Adam:




雖然validation loss在訓練過程中一直上升，但也沒有overfitting的現象，因為training loss也沒有到很低。

最後有試一個巨大的model(Github上的model):
Encoder: 2-layer bidirectional LSTM(hidden units=512)
Decode: 2-layer bidirectional LSTM(hidden units=1024)，加上Luong Attention。

結果為

| | Greedy | Beam |
|---|---|---|
| Perplexity | 11.02 | 10.52 |
| Ccorrelation | 0.7 | 0.67 |