Generating Text with Deep Reinforcement Learning, H. Guo et al, 2015.

Lee Won-ho
gh9908@gmail.com

Index

- Prerequisites
- Paper main concepts
- Experiment

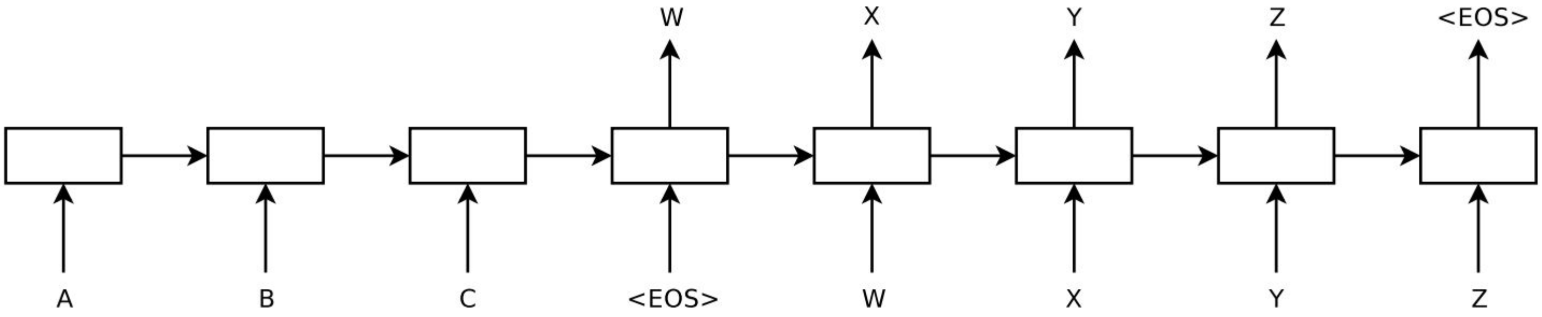
Generating Text with Deep Reinforcement Learning, H. Guo et al, 2015.

Prerequisites

- s2s learning
- Bidirectional LSTM
- Gradient Clipping
- DQN
- Replay Memory Sampling

Sequence-to-Sequence Learning

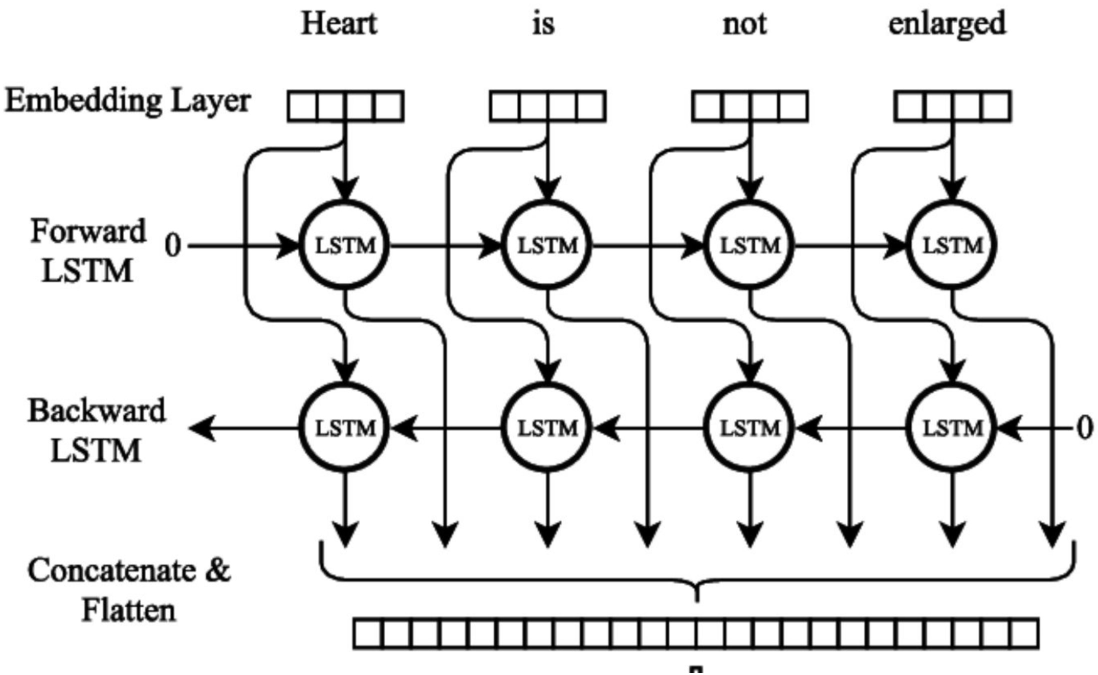
- language model used in sequence prediction tasks
- one LSTM - encoder (read input sequence)
- another LSTM - decoder (extract the output sequence)



Generating Text with Deep Reinforcement Learning, H. Guo et al, 2015.

Bidirectional LSTM

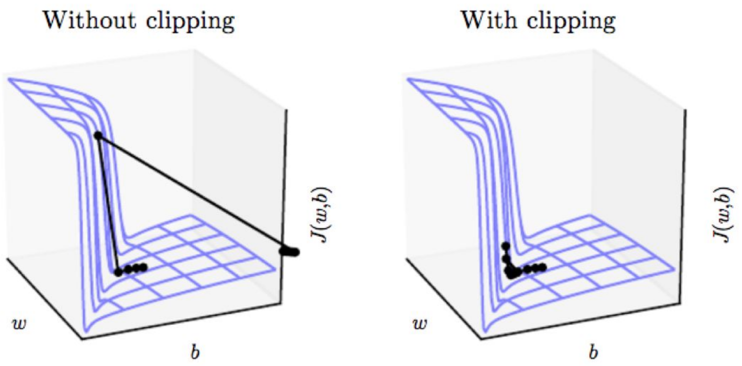
- sequence processing model that consists of two LSTMs
- one LSTM - forward direction , the other LSTM - backwards direction
- model can know whole information of the sequences



Generating Text with Deep Reinforcement Learning, H. Guo et al, 2015.

Gradient Clipping

- two issues with properly training RNN
  - Vanishing Gradient
  - Exploding Gradient
- clip the norm **|| g ||** of the gradient **g** before a parameter update



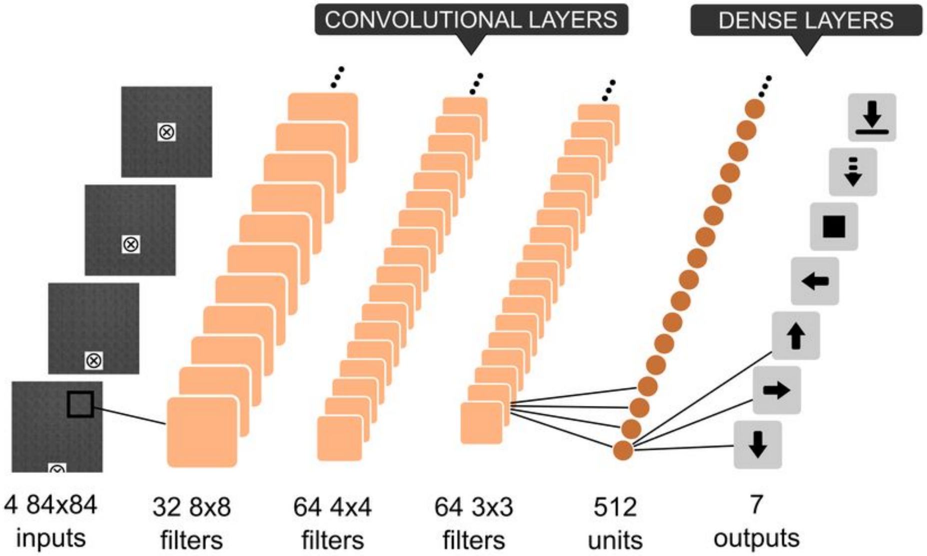
$$\text{if } \|\mathbf{g}\| > v \text{ then } \mathbf{g} \leftarrow \frac{\mathbf{g}^v}{\|\mathbf{g}\|}$$

[On the difficulty of training Recurrent Neural Networks, R.Pascanu et al,2013]

Generating Text with Deep Reinforcement Learning, H. Guo et al, 2015.

DQN

- approximates a state-value function in a Q-Learning framework with a neural network



Generating Text with Deep Reinforcement Learning, H. Guo et al, 2015.

Replay Memory Sampling

- store the agent's experiences at each time-step in a data-set pooled over many episodes into a replay memory and then, sample the memory

- Experience Replay
  - randomly sample in memory pool for a minibatch
- Prioritized Experience Replay
  - measured by the magnitude of TD error
  - stochastic prioritization

[Prioritized Experience Replay, T.Schaul et al, 2016]

Generating Text with Deep Reinforcement Learning, H. Guo et al, 2015.

Abstract

- The **aim** here is to enable the **decoder to first tackle easier portions** of the sequences, and then turn to cope with difficult parts
- encoder-decoder is employed to automatically create features to represent the **internal states** of and formulate a list of potential **actions** for the DQN
- The DQN learns to make decision on which action will be selected from the list to **modify the current decoded sequence**.
- compared to a left-to-right greedy beam search LSTM decoder, the proposed method **outperformed the baseline** when decoding unseen sentences

Generating Text with Deep Reinforcement Learning, H. Guo et al, 2015.

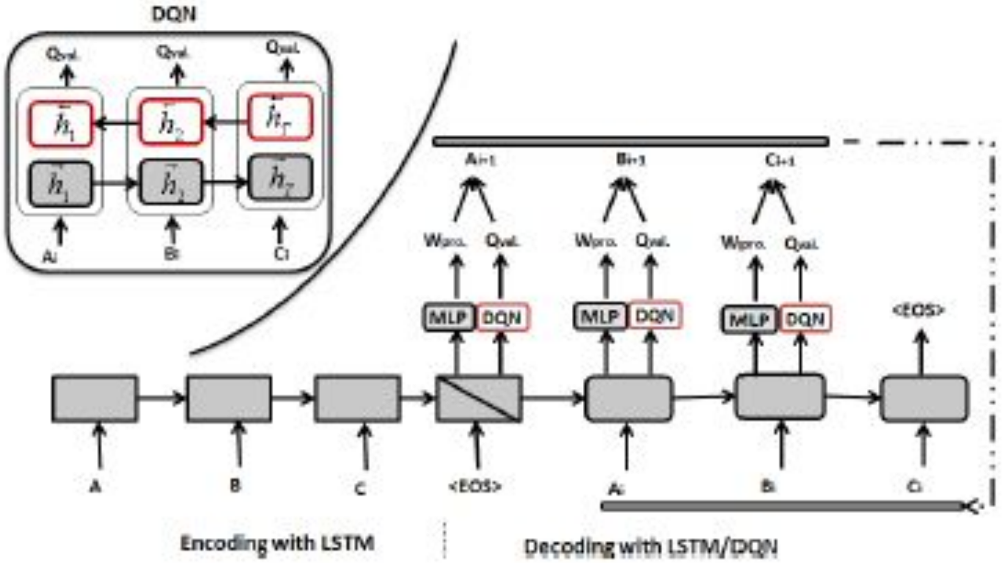
Algorithm

Algorithm 1 Generating Text with Deep Q-Network

Initialize replay memory D; initialize EnLSTM, DeLSTM, and DQN with random weights
Pretraining Encoder-Decoder LSTMs
for epoch = 1,M **do**
    randomize given training set with sequence pairs $< X, Y >$.
    for each sequence pair $EnSen^k \in X$ and $TaSen^k \in Y$ **do**
        Encode $EnSen^k$ with EnLSTM, and then predict the next token (e.g., word) in $TaSen^k$ with DeLSTM.
    end for
end for
Training Q-value function
for epoch = 1,U **do**
    for each sequence pair $EnSen^k \in X$ and $TaSen^k \in Y$ (with length l) **do**
        feed $EnSen^k$ into pretrained encoder-decoder LSTMs; obtain the decoded sequence $DeSen_0^k$
        for iteration i = 1, $2l$ **do**
            if random() $< \epsilon$ **then**
                select a random action a_t (e.g., word w) at time step t of $DeSen_i^k$ (selection biases to incorrect decoded tokens)
            else
                compute $Q(s_i, a)$ for all actions using DQN; select $a_t = argmaxQ(S_i, a)$, resulting in a new token w for the t-th token in $DeSen_i^k$
            end if
            replace $DeSen_i^k$ with w, resulting $DeSen_{i+1}^k$
            compute the similarity of $DeSen_{i+1}^k$ and $TaSen^k$, resulting reward score r_i
            store transition tuple $[s_i, a_i, r_i, s_{i+1}]$ in replay memory D; $s_i = [EnSen_i^k, DeSen_i^k]$.
            random sample of transition $[s_i, a_t, r_i, s_{i+1}]$ in D
            if $r_i > \sigma$ (preset BLEU score threshold) **then**
                $q_i = r_i$; current sequence decoding successfully complete.
            else
                $q_i = r_i + \lambda max_{a'} Q(s', a'; \theta_{i-1})$
            end if
            perform gradient descent step on only the DQN network $(q_i - Q(s, a; \theta_i))^2$
        end for
    end for
end for

StateGF (State Generation Function)

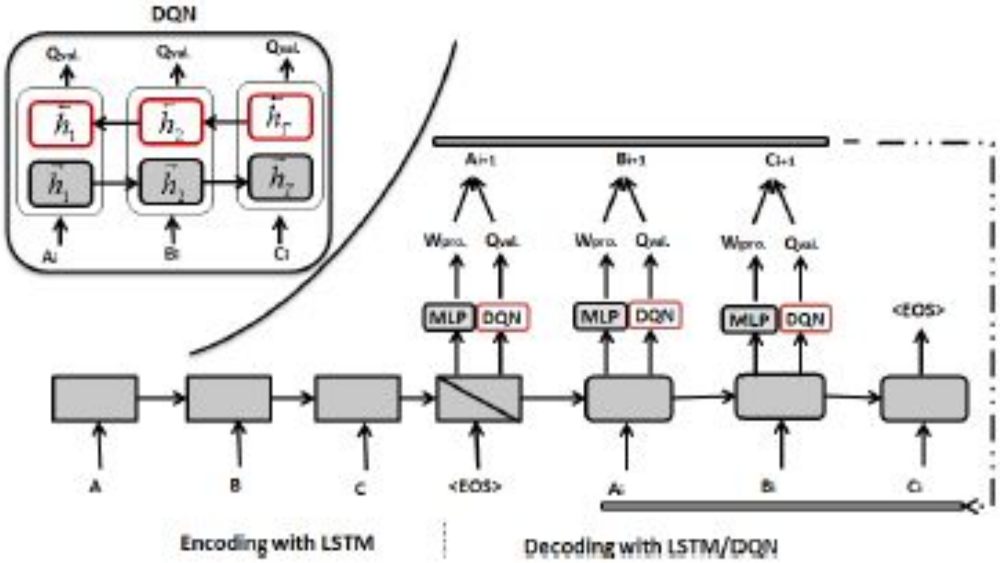
- The encoder-decoder LSTM network is depicted as gray-filled rectangles in Figure 1. For descriptive purpose, we named this State Generation Function (denoted as StateGF) under the context of DQN



Generating Text with Deep Reinforcement Learning, H. Guo et al, 2015.

StateGF (State Generation Function)

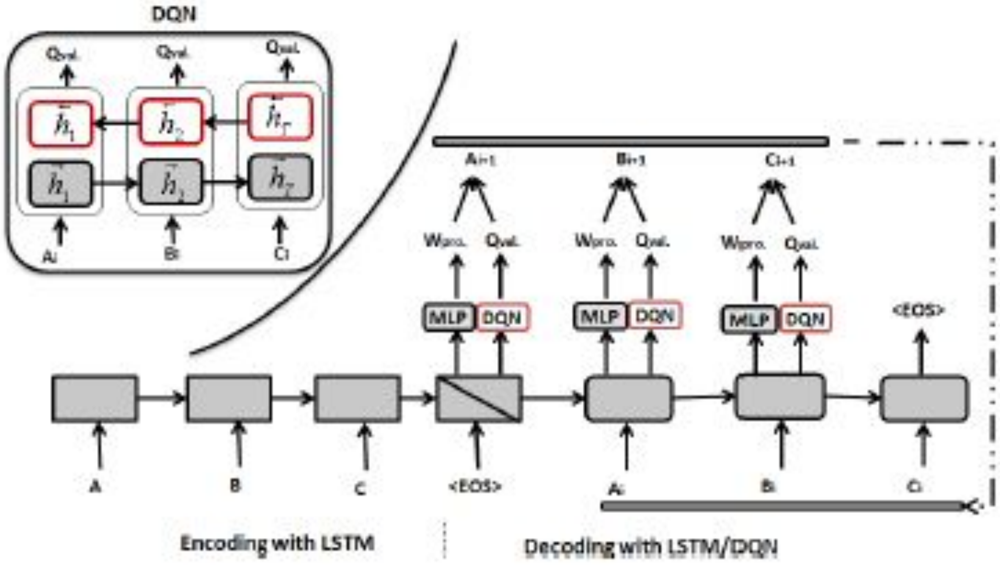
- encode the sequence using one LSTM (denoted as EnLSTM), reading into the tokens one timestep at a time
- this encode process results in a fixed dimensional vector representation for the whole sentence h_N^{en}



Generating Text with Deep Reinforcement Learning, H. Guo et al, 2015.

StateGF (State Generation Function)

- the resulted vector is used as the initial state of another LSTM (denoted as DeLSTM) for decoding to generate the target sequence



Generating Text with Deep Reinforcement Learning, H. Guo et al, 2015.

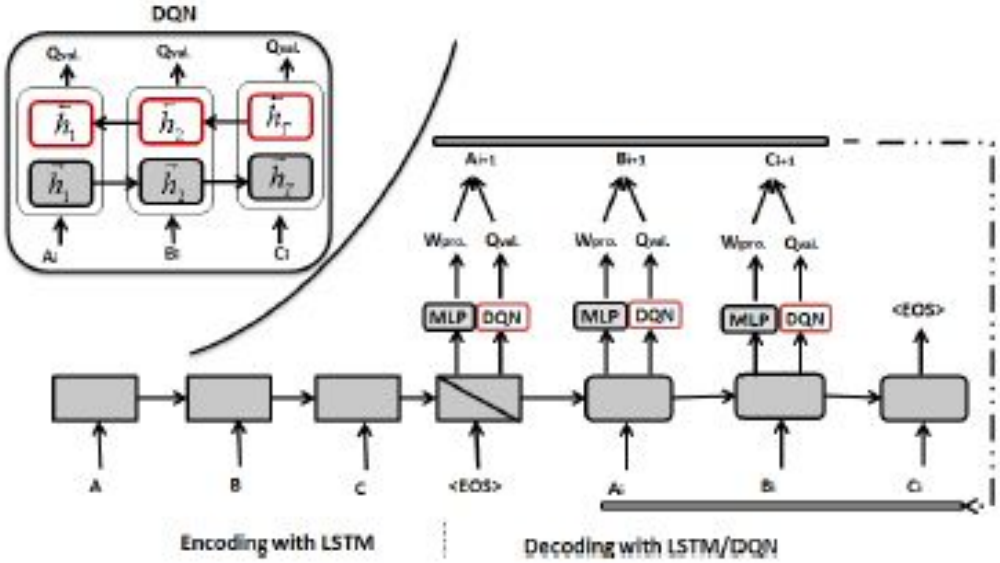
StateGF (State Generation Function)

- the DeLSTM creates a sequence of hidden state for each time step
- each of these hidden vectors is fed into a Softmax function to produce a distribution over the C possible classes (e.g., words in a vocabulary or dictionary)

$$P(W_{pro}^{t} = c | EnSen, \vartheta) = \frac{exp(w_c^T h_t^{de})}{\sum_{c=1}^{C} exp(w_c^T h_t^{de})}$$

$$1/|S| \sum_{(X,Y)\in S} logp(Y|X)$$

$$\hat{Y} = \underset{Y}{\operatorname{argmax}} p(Y|X)$$



BLEU Score for DQN Reward

- Reward is calculated based on the closeness between the target sentence and the decoded output sentence after the DQN takes an action
- measure the score difference between the current iteration and the previous iteration

Generating Text with Deep Reinforcement Learning, H. Guo et al, 2015.

Empirical Observations on Model Design

- Separating State Generation Function from DQN
- Pre-training the State Generation Function
- Updating with Replay Memory Sampling
- Importance of Supervised Softmax Signal
- Simultaneously Updating with Both Softmax and Q-value

Generating Text with Deep Reinforcement Learning, H. Guo et al, 2015.

Separating StateGF from DQN

- Using DQN to approximate the Q-value function equals to train a network against moving targets because here the network's targets depend on the network itself
- Suppose, for a given input feed, the StateGF would generate a different output sequence each time for the DQN
- the DQN network has to also deal with a moving state function involving text with ***very high dimensionality***

Generating Text with Deep Reinforcement Learning, H. Guo et al, 2015.

Pre-training the StateGF

- two empirical techniques are employed to ensure that we have a deterministic network for generating states for DQN
    - pre-training StateGF
    - iteratively DeSen is used by the DeLSTM as next input

Generating Text with Deep Reinforcement Learning, H. Guo et al, 2015.

Updating with Replay Memory Sampling

- Our studies also indicate that, performing updates to the Q-value function using transitions from the current training sentence causes the network to stronglty **overfit the current input sentence**.
- Transition Tuple

$$[(\text{EnSen}_i, \text{DeSen}_i), \widehat{y_t^i}, r_i, ([\text{EnSen}_i, \text{DeSen}_{i+1}])$$

Generating Text with Deep Reinforcement Learning, H. Guo et al, 2015.

Importance of Supervised Softmax Signal

- the whole network, including the LSTMs and DQN, only receive the error signals from the Q-value predictions
- **without the supervised signal** the DQN was very **difficult to learn.**
    - moving StateGF and a moving Q-value target function
    - word probability list for each output of the DeLSTM is changing and unreliable

Simultaneously Updating with Both Softmax and Q-value Error

Generating Text with Deep Reinforcement Learning, H. Guo et al, 2015.

- If during training the DQN, we not only update the DQN as discussed previously, but also update the state generation functions.
- We found that the network could be **easily bias to the state generation functions** since the Softmax error signal is very strong and more reliable
- we could bias towards the learning of DQN, but this would introduce one more **tricky parameter for tuning**, then we have an **indeterministic** state generation function again

Generating Text with Deep Reinforcement Learning, H. Guo et al, 2015.

Dataset

- randomly select 12,000 sentences, with max length of 30, from the Billion Word Corpus
- for train 10,000 sentences
- for validation 1,000 sentences
- for test 1,000 sentence from training set, 1,000 unseen sentences

Training and Testing Detail

- For each sentence with length of l , we allow DQN to edit the sentence with $2l$ iterations
- Since the maximal length of a sentence in our experiment is 30, the DQN has at most 31 output nodes.
- the DQN can choose one of the 30 top words, each corresponding to a time step at the DeLSTM, as its action, or take the 31st action which indicates not modification is needed

Experimental Results

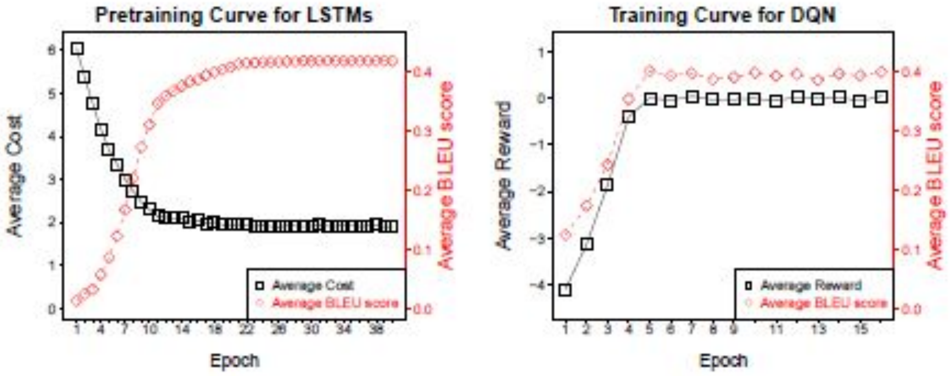


Figure 2: The evolution of cost for training the StateGF and reward for training the DQN.

Testing systems	LSTM decoder	DQN
Average SmoothedBLEU on sentences IN the training set	0.425	0.494
Average SmoothedBLEU on sentences NOT in the training set	0.107	0.228

Table 1: Experimental results for decoding the seen and unseen sentences in testing.

Generating Text with Deep Reinforcement Learning, H. Guo et al, 2015.

Experimental Results