In this homework, we will summarize two papers related to NN with memory

The papers include:

**1. LSTM: A Search Space Odyssey**

**2. Dynamic Neural Turing Machine with Soft and Hard Addressing Schemes**

負責部份:

1. LSTM: A Search Space Odyssey:

Organizer: 周育潤, 翁慶年

2. Dynamic Neural Turing Machine with Soft and Hard Addressing Schemes

Organizer: 鄭乃嘉, 賴筱婷

In general, the AI tasks like speech recognition, language modeling or machine translation are very difficult for RNN to handle due to the long term dependencies, which will induce vanishing and exploding gradients.

In theory, RNN can use past information as new input while in practice, RNN can't really learn well when related information are separated far apart.

To tackle above challenge, one approach is to focus on gradient part and develop different optimization method like Hessian-Free, adaptive learning rates or stochastic gradient with momentum.

Another approach is to focus on design a small system to memorize something since the input data has long term dependencies.

The original idea of "memory system", LSTM, is introduced by Hochreiter & Schmidhuber (1997).

LSTM achieved impressive result in many AI tasks and hence generate lots of variants like GRU(Gated Recurrent Unit), peephole connection, Depth Gated RNN and etc.

Since lots of variants are proposed, it's very important to figure out which part of

the complex LSTM design contribute to the success.

At Paper-1, **LSTM: A Search Space Odyssey** do a comprehensive split testing of eight variants of vanilla LSTM

The structure of vanilla LSTM is illustrated as below

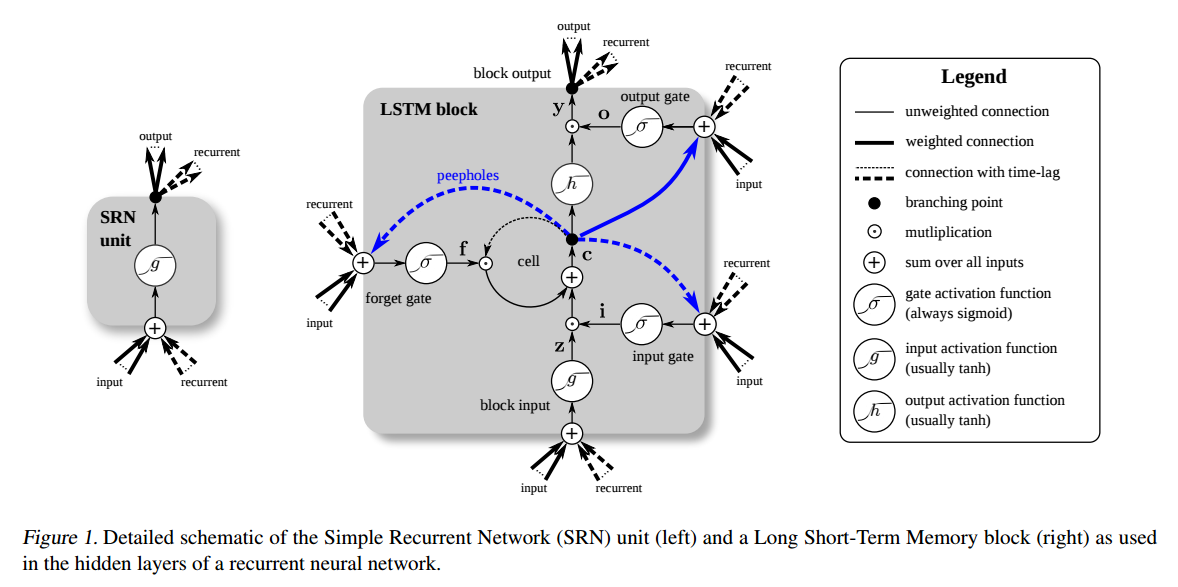


Fig 1. Illustration of vanilla LSTM

This paper compare eight derived variants of LSTM which consist of:

1. No Input Gate (NIG)

2. No Forget Gate (NFG)

3. No Output Gate (NOG)

4. No Input Activation Function (NIAF)

5. No Output Activation Function (NOAF)

6. No Peepholes (NP)

7. Coupled Input and Forget Gate (CIFG)

8. Full Gate Recurrence (FGR)

The variants are tested on three benchmark tasks:

acoustic modeling, handwriting recognition and polyphonic modeling.

To compare these variants fairly, the hyper-parameters of each task were optimized by random search and their importance is evaluated by faNOVA framework

They have several important conclusion

1. The performance of these eight variants are basically the same

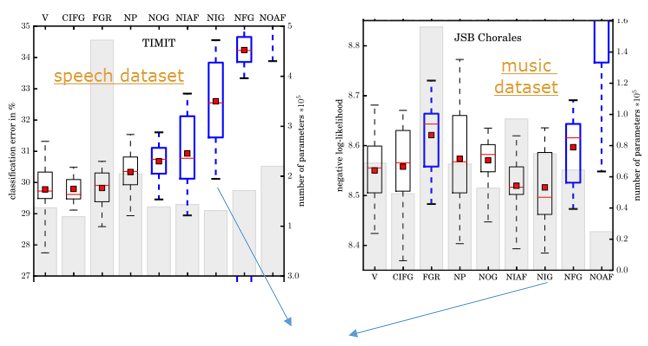
2. In general, forget gate and output activation function are most critical for LSTM

3. The effect of input gate or output gate depends on the task being applied

Their results provide very helpful insight for us to choose proper LSTM variants when we apply NN with memory on a new task

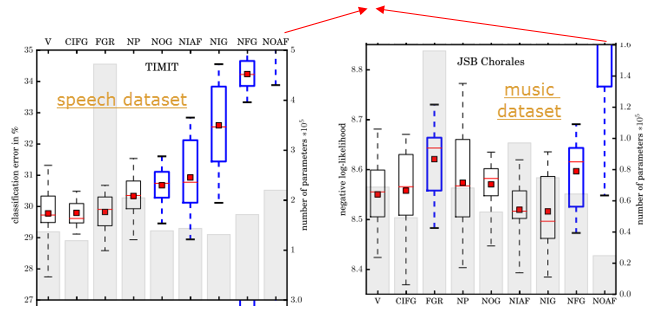
For example,

1. **NIG (No input gate) has obvious impact TIMIT(speech) while no impact on JSB(music)**
2. **No Output Activation function (NOAF) will seriously suffer all the tasks**



**NIG (No input gate) has obvious impact TIMIT(speech) while no impact on JSB(music)**

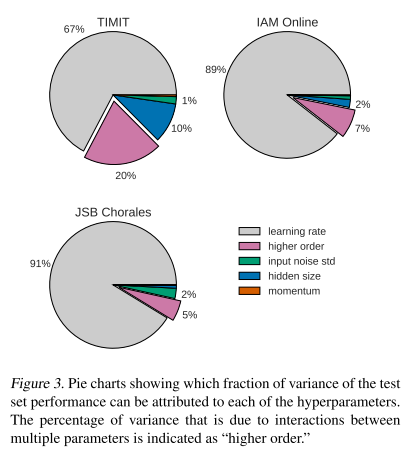
**No Output Activation function (NOAF) will seriously suffer all the tasks**



Besides the conclusive result, this paper also utilizes one powerful framework, faNOVA, to identify the importance of different model parameters.

I am still trying to catch this interesting faNOVA technique by google and wish to understand more clear about the main idea.

With faNOVA, this paper point out that importance priority of model tuning parameters will be learning-rate >> higher order interactions > hidden layer size > input noise > momentum like below picture.



Since parameter of learning rate cause the major variance while other parameter like high order interaction seems small when compared to it, author suggest that we can use a small network to tune the rate first to save experimentation time.

That’s very practical suggestion of tuning model from faNOVA framework result.

In sum, this paper provides a systematic study for us to get a clear overview of different LSTM structure and gain insight like

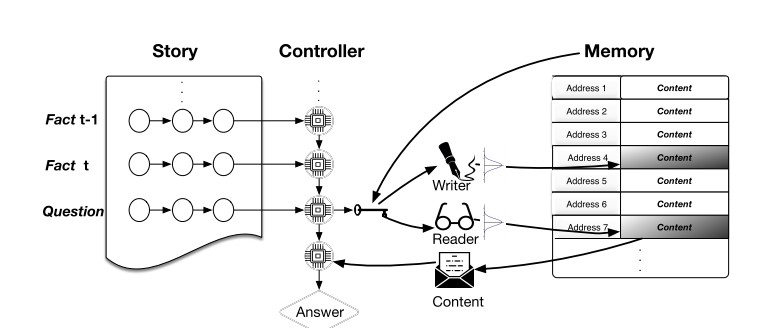
1. Removing peephole or coupling input & forget gate does not hurt the performance seriously
2. Output activation function and forget gate are crucial
3. Learning rate is the most important model parameters and can be tuned with small network

Now, let’s pay our attention to paper-2, **Dynamic Neural Turing Machine with Soft and Hard Addressing Schemes**

The memory concept in LSTM and its variants basically is implicit memory (or internal memory) which is in the form of recurring hidden states. The memory concept here actually is not so straightforward for us to understand.

On the contrary, the explicit (or external) memory concept in neural Turing machine is clear and it is just kinds of similar to the memory of a laptop.

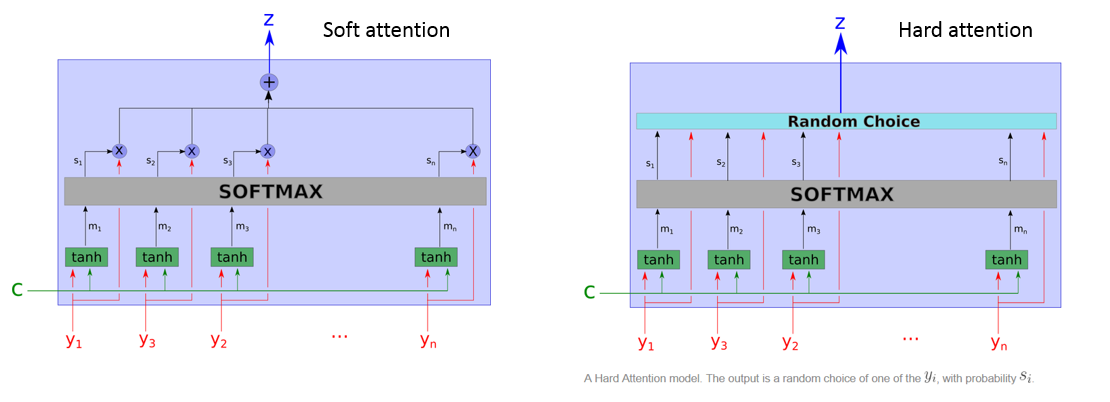
To access these explicit memory, the network, whether feed-forward or RNN, will learn to **address** the memory cell. That means the network is like a controller learning to read/write the memory cell as below



This paper introduces a trainable addressing scheme which utilize two separate vectors, content and address vectors for each memory cell. The address vectors are considered a model parameter and being update during training.

For addressing strategy, both soft attention and hard attention are being tested and the result shows that hard-attention with GRU controller performs better than soft attention.

The detail of hard-soft attention model is not explained at this paper while it’s worth to add the information here for deeper understanding



Soft attention is differentiable and can be integrated into existing network since the gradients are propagated through attention mechanism the same way it through the network. It will use all the hidden state as input for decoding.

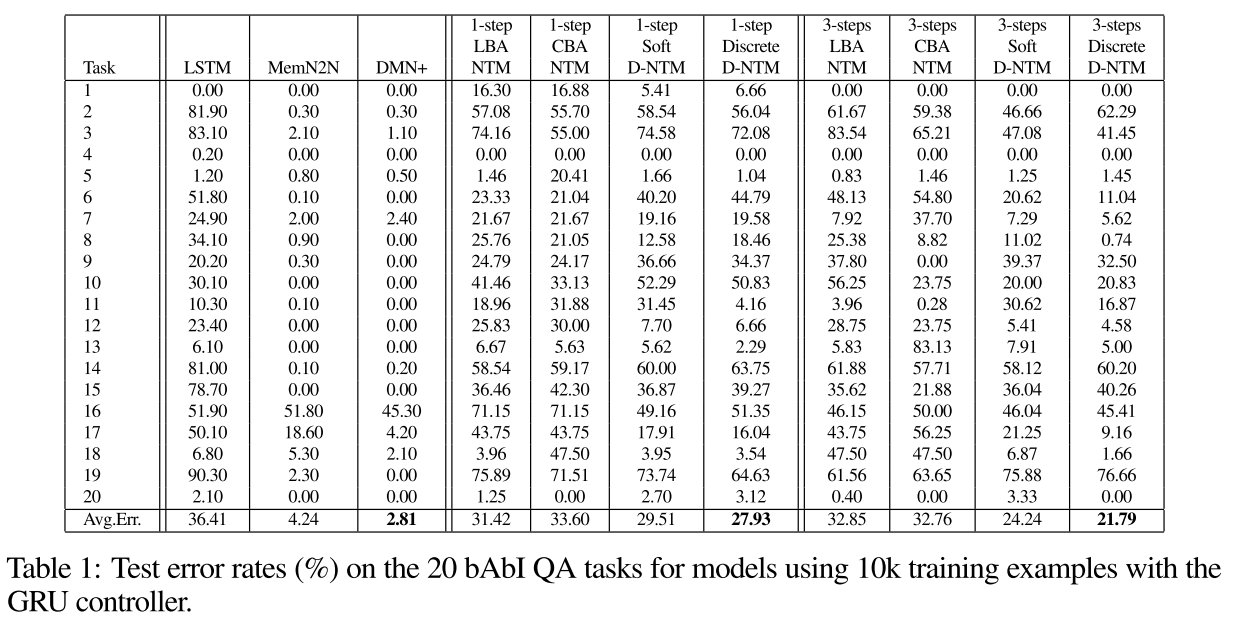
Hard attention is a stochastic process which sample the hidden state with probability.

To propagate the gradient with random process, Monte Carlo sampling is used to estimate the gradient.

Actually, the proposed D-NTM (dynamic neural Turing machine) framework at this paper does not show the state-of-art result compared to other Memory network like end-to-end Memory network(MemN2N). However, the author suggests that D-NTM has the potential to perform better when tasks involve huge data which make it difficult to memorize everything and the difficulty will challenge the performance of Memory network like MemN2N.

As below picture, we can see the framework of this paper **D-NTM** may be better than

**LSTM** but still far beyond the performance of Memory network like **MemN2N**



In sum, this paper still provides us a very valuable reference for us to design neural network with memory using NTM and attention model.

We can do more extend study to check whether any enhancement to boost the D-NTM framework.