LSTM and GRU on Language Model

Tianyi Chen, Zhouyang Zhang

May 9, 2016

Abstract

In this project, we derive and implement two variants of RNNs, namely Long Short Term Memory (LSTM) and Gated Recurrent Units (GRU). Comparing to standard RNNs, both of them are superior in memorizing and filtering long time lag information. We train both of them on a 10000 English sentence dataset from [6] to build a language model. Based on Numerical experiments, cross entropy loss value of both LSTM and GRU decrease. Furthermore, we build a reranker based on GRU, which can achieve similar performance comparing to given n-gram count collecting reranker. Thus, GRU language model may be a competitive method to traditional language model in natural language processing.

1 Recurrent Neural Network

Traditional feedforward neural networks often assume the input data are independent among others, but this property is unsuitable for many tasks in natural language processing, like language model. For example, consider a sentence “The color of this apple is red”, consisting of seven words, and each word is relative to others. If one person saw the words “color” and “apple”, it is natural for this person to expect the coming word as “red” or “green”. To complete many tasks in language process, we can not assume the independency among data, extra structure is needed to store previous information.

Recurrent Neural Networks (RNNs) is a neural networks that is particularly good at dealing sequential inputs. As traditional feedforward neural networks, RNNs take the input data as input layer, and forward propagate the input to the output layer through hidden layers. But RNNs also take additional information produced by previous conditions with “memory” for saving previous information. Such “memory” also causes the difference in training RNNs, instead of standard backpropagation, backpropagation through time is used for training RNNs.

The generic “unfolded” structure of RNNs is shown in Figure 1. There exist many variants of RNNs, two of them are LSTM and GRU with special designed “gates” structures. In the following sections, we will show details of these two variants.

1.1 Notations

The following notations will occur in the rest sections:

1. $a \circ b$: vector a multiply vector b elementwise;
2. $a \otimes b$: outer product of vector a and b;
3. $\sigma(x)$: sigmoid function value of vector x elementwise;
4. $softmax(x)$: softmax function value of vector x elementwise;
5. $tanh(x)$: tanh function value of vector x elemenwise

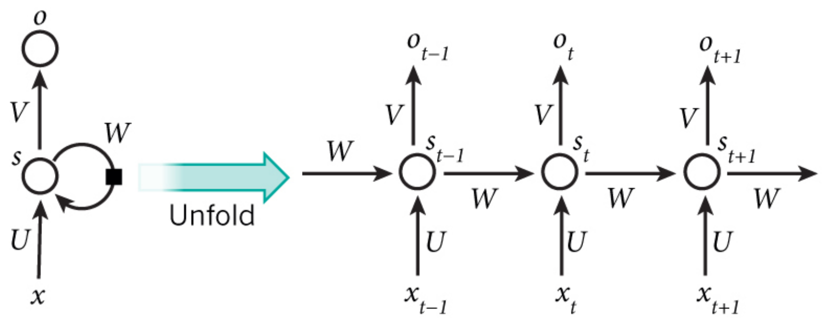

Figure 1: Standard Recurrent Neural Network

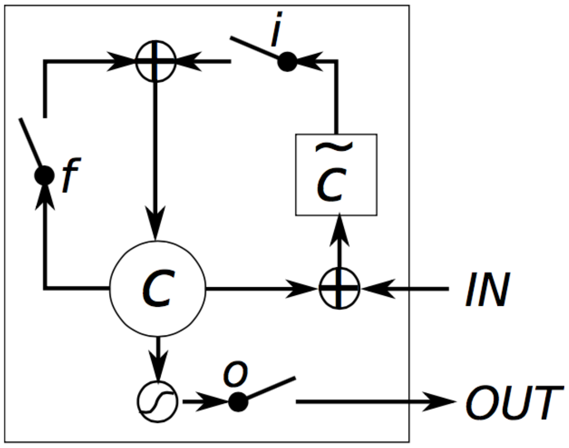

Figure 2: LSTM, units

1.2 Long Short Term Memory (LSTM)

Back-propagated error in standard RNNs quickly either vanishes or blows up[3], which cause it fails to learn in the presence of long time lags. Long Short Term Memory is introduced by Hochreiter and Schmidhuber[2] to overcome this problem by enforcing non-decaying error flow through "Constant Error Carrousels" within special units, called cells. However, traditional LSTM fails to learn to correctly process certain very long or continual time series problems, since a continual input stream eventually may cause the internal values of cells to grow without bound. To handle this issue, forget gate is introduced as a remedy to reset cell rhythmically, and consist of current LSTM structure.

As shown in Figure 2, LSTM contains the gate i responsible for acception/rejection of new input, gate f responsible for combining new/old memory, and gate o for determining the output.

1.2.1 Forward Propogation

Let $x(t)$ be the input in time step t (i.e. the $t-th$ word), $s(t)$ be the hidden state of $x(t)$, and $\hat{y}(t)$ be the output for $x(t)$. Then the forward propagation of LSTM can be shown in the belows:

$$i(t) = \sigma(U^i x(t) + W^i s(t-1)) \tag{1}$$
$$f(t) = \sigma(U^f x(t) + W^f s(t-1)) \tag{2}$$
$$o(t) = \sigma(U^o x(t) + W^o s(t-1)) \tag{3}$$
$$g(t) = tanh(U^g x(t) + W^g s(t-1)) \tag{4}$$
$$c(t) = c(t-1) \circ f(t) + g(t) \circ i(t) \tag{5}$$
$$s(t) = c(t) \circ o(t) \tag{6}$$
$$\hat{y}(t) = softmax(V s(t)) \tag{7}$$

1.2.2 Backpropogation Through Time

As training RNNs, backpropogation through time can be still used to train LSTM. However, due to complicated structure, it is difficult to derive the exact Backpropogation Through Time on LSTM. People prefer to use numeral methods to approximate the gradient over each variable, like Theano's *grad()* function. Instead of using numerical method to estimate gradients, we derive the whole backpropogation through time on LSTM exactly.

Denote $l(t)$ as the loss function for $x(t)$, L as the loss over the whole dataset, and select cross entropy loss, that is

$$l(t) = -y(t) \cdot \log(\ \hat{y}(t)\ ) \tag{8}$$
$$L = \sum_{t=1}^{T} l(t) \tag{9}$$

Also introduce $L(t)$ to represents the cumulative loss from step t onwards:

$$L(t) = \sum_{j=t}^{T} l(j)$$

And assume $\sum_{j=1}^{T} y(j) = 1$. Now we present the gradients of L over each component with deduction:

1. Gradient of L over $\hat{y}(t)$:

   It is trivial that

$$\begin{aligned}\frac{\partial L}{\partial \hat{y}(t)} &= \sum_{j=1}^{T} \frac{\partial l(j)}{\partial \hat{y}(t)} \\ &= \frac{\partial l(t)}{\partial \hat{y}(t)} = -\frac{y(t)}{\hat{y}(t)}\end{aligned} \tag{10}$$

2. Gradient of L over $z(t) = V s(t)$:

   With the assumption $\sum_{j=1}^{T} y(j) = 1$, we have

$$
\begin{aligned}
\frac{\partial L}{\partial z(t)} &= \sum_{j=1}^{T} \frac{\partial l(j)}{\partial \hat{y}(j)} \frac{\partial \hat{y}(j)}{\partial z(t)} \\
&= \sum_{j=1}^{T} -\frac{y(j)}{\hat{y}(j)} \frac{\partial \hat{y}(j)}{\partial z(t)} \\
&= -\frac{y(t)}{\hat{y}(t)} \hat{y}(t)(1-\hat{y}(t)) + \sum_{j\neq t}^{T} \frac{y(j)}{\hat{y}(j)} \hat{y}(j)\hat{y}(t) \\
&= -y(t)(1-\hat{y}(t)) + \sum_{j\neq t}^{T} y(j)\hat{y}(t) \\
&= -y(t) + \hat{y}(t)y(t) + \hat{y}(t) \sum_{j\neq t}^{T} y(j) \\
&= -y(t) + \hat{y}(t) \sum_{j=1}^{T} y(j) \\
&= \hat{y}(t) - y(t) \qquad (11)
\end{aligned}
$$

3. Gradient of L over V:

   By chain rule, $z(t) = Vs(t)$ and (11):

$$
\begin{aligned}
\frac{\partial L}{\partial V} &= \sum_{j=1}^{T} \frac{\partial L}{\partial z(j)} \frac{\partial z(j)}{\partial V} \\
&= \sum_{j=1}^{T} (\hat{y}(j) - y(j)) \cdot \frac{\partial z(j)}{\partial V} \\
&= \sum_{j=1}^{T} (\hat{y}(j) - y(j)) \otimes s(j) \qquad (12)
\end{aligned}
$$

4. Gradient of L over $s(t)$:

   By the definition of $s(t)$, $s(t)$ is relative to $l(t)$, and $l(t+1)$, then

$$
\frac{\partial L}{\partial s(t)} = \frac{\partial l(t)}{\partial s(t)} + \frac{\partial l(t+1)}{\partial s(t)} \qquad (13)
$$

   where by chain rule for vectors, $z(t) = Vs(t)$ and (11):

$$
\begin{aligned}
\frac{\partial l(t)}{\partial s(t)} &= \frac{\partial z(t)}{\partial s(t)} \frac{\partial l(t)}{\partial z(t)} \\
&= \frac{\partial z(t)}{\partial s(t)} \cdot (\hat{y}(t) - y(t)) \\
&= V^T (\hat{y}(t) - y(t)) \qquad (14)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial l(t+1)}{\partial s(t)} &= \frac{\partial i(t)}{\partial s(t)} \frac{\partial l(t+1)}{\partial i(t)} + \frac{\partial f(t)}{\partial s(t)} \frac{\partial l(t+1)}{\partial f(t)} + \frac{\partial o(t)}{\partial s(t)} \frac{\partial l(t+1)}{\partial o(t)} + \frac{\partial g(t)}{\partial s(t)} \frac{\partial l(t+1)}{\partial g(t)} \\
&= [W^i]^T \frac{\partial i(t)}{\partial s(t)} + [W^f]^T \frac{\partial f(t)}{\partial s(t)} + [W^o]^T \frac{\partial o(t)}{\partial s(t)} + [W^g]^T \frac{\partial g(t)}{\partial s(t)} \qquad (15)
\end{aligned}
$$

Hence to calculate $\frac{\partial L}{\partial s(t)}$, we need to calculate $\frac{\partial l(T)}{\partial s(T)}$ at first, and calculate (15) during the backpropogation.

5. Gradient of L over $c(t)$:

By the definition of $c(t)$, $c(t)$ is relative to $s(t)$, and $c(t+1)$, then by chain rule for vectors, and (13):

$$\begin{aligned}\frac{\partial L}{\partial c(t)} &= \frac{\partial L}{\partial s(t)}\frac{\partial s(t)}{\partial c(t)} + \frac{\partial L}{\partial c(t+1)}\frac{\partial c(t+1)}{\partial c(t)} \\ &= \frac{\partial L}{\partial s(t)} \circ o(t) + \frac{\partial L}{\partial c(t+1)} \circ f(t) \end{aligned} \tag{16}$$

Notice $\frac{\partial L}{\partial c(T+1)} = 0$, since $c(T+1)$ does not exist. Hence, we can iteratively calculate

$$\frac{\partial L}{\partial c(T)}, \frac{\partial L}{\partial c(T-1)}, \cdots, \frac{\partial L}{\partial c(t)}$$

in backpropogation process.

6. Gradient of L over $o(t)$:

By the definition $o(t)$ and (13), we have

$$\begin{aligned}\frac{\partial L}{\partial o(t)} &= \frac{\partial L}{\partial s(t)}\frac{\partial s(t)}{\partial o(t)} \\ &= \frac{\partial L}{\partial s(t)} \circ c(t) \end{aligned} \tag{17}$$

7. Gradient of L over $i(t)$:

By the definition $i(t)$ and (16), we have

$$\begin{aligned}\frac{\partial L}{\partial i(t)} &= \frac{\partial L}{\partial c(t)}\frac{\partial c(t)}{\partial i(t)} \\ &= \frac{\partial L}{\partial c(t)} \circ g(t) \end{aligned} \tag{18}$$

8. Gradient of L over $f(t)$:

By the definition $f(t)$ and (16), we have

$$\begin{aligned}\frac{\partial L}{\partial f(t)} &= \frac{\partial L}{\partial c(t)}\frac{\partial c(t)}{\partial f(t)} \\ &= \frac{\partial L}{\partial c(t)} \circ c(t-1) \end{aligned} \tag{19}$$

9. Gradient of L over $g(t)$:

By the definition $g(t)$ and (13), we have

$$\begin{aligned}\frac{\partial L}{\partial g(t)} &= \frac{\partial L}{\partial s(t)}\frac{\partial s(t)}{\partial g(t)} \\ &= \frac{\partial L}{\partial s(t)} \circ i(t) \end{aligned} \tag{20}$$

10. Gradient of L over U^i:

By the definition U^i and (18), we have

$$\begin{aligned}\frac{\partial L}{\partial U^i} &= \sum_{j=1}^{T} \frac{\partial l(j)}{\partial i(j)} \frac{\partial i(j)}{\partial U^i x(j)} \frac{\partial U^i x(j)}{\partial U^i} \\ &= \sum_{j=1}^{T} \frac{\partial l(j)}{\partial i(t)} \circ i(j) \circ (1 - i(j)) \otimes x(j) \end{aligned} \tag{21}$$

11. Gradient of L over U^f:

By the definition U^f and (19), we have

$$\begin{aligned}\frac{\partial L}{\partial U^f} &= \sum_{j=1}^{T} \frac{\partial l(j)}{\partial f(j)} \frac{\partial f(j)}{\partial U^f x(j)} \frac{\partial U^f x(j)}{\partial U^f} \\ &= \sum_{j=1}^{T} \frac{\partial l(j)}{\partial f(t)} \circ f(j) \circ (1 - f(j)) \otimes x(j) \end{aligned} \tag{22}$$

12. Gradient of L over U^o:

By the definition U^o and (17), we have

$$\begin{aligned}\frac{\partial L}{\partial U^o} &= \sum_{j=1}^{T} \frac{\partial l(j)}{\partial o(j)} \frac{\partial o(j)}{\partial U^o x(j)} \frac{\partial U^o x(j)}{\partial U^o} \\ &= \sum_{j=1}^{T} \frac{\partial l(j)}{\partial o(t)} \circ o(j) \circ (1 - o(j)) \otimes x(j) \end{aligned} \tag{23}$$

13. Gradient of L over U^g:

By the definition U^g and (20), we have

$$\begin{aligned}\frac{\partial L}{\partial U^g} &= \sum_{j=1}^{T} \frac{\partial l(j)}{\partial g(j)} \frac{\partial g(j)}{\partial U^g x(j)} \frac{\partial U^g x(j)}{\partial U^g} \\ &= \sum_{j=1}^{T} \frac{\partial l(j)}{\partial g(t)} \circ (1 + g(j)) \circ (1 - g(j)) \otimes x(j) \end{aligned} \tag{24}$$

14. Gradient of L over W^i:

By the definition W^i and (18), we have

$$\begin{aligned}\frac{\partial L}{\partial W^i} &= \sum_{j=1}^{T} \frac{\partial l(j)}{\partial i(j)} \frac{\partial i(j)}{\partial W^i s(j-1)} \frac{\partial W^i s(j-1)}{\partial W^i} \\ &= \sum_{j=1}^{T} \frac{\partial l(j)}{\partial i(t)} \circ i(j) \circ (1 - i(j)) \otimes s(j-1) \end{aligned} \tag{25}$$

15. Gradient of L over W^f:

By the definition W^f and (19), we have

$$\begin{aligned}\frac{\partial L}{\partial W^f} &= \sum_{j=1}^{T} \frac{\partial l(j)}{\partial f(j)} \frac{\partial f(j)}{\partial W^f s(j-1)} \frac{\partial W^f s(j-1)}{\partial W^f} \\ &= \sum_{j=1}^{T} \frac{\partial l(j)}{\partial f(t)} \circ f(j) \circ (1 - f(j)) \otimes s(j-1) \qquad (26)\end{aligned}$$

16. Gradient of L over W^o:

By the definition W^o and (17), we have

$$\begin{aligned}\frac{\partial L}{\partial W^o} &= \sum_{j=1}^{T} \frac{\partial l(j)}{\partial o(j)} \frac{\partial o(j)}{\partial W^o s(j-1)} \frac{\partial W^o s(j-1)}{\partial W^o} \\ &= \sum_{j=1}^{T} \frac{\partial l(j)}{\partial o(t)} \circ o(j) \circ (1 - o(j)) \otimes s(j-1) \qquad (27)\end{aligned}$$

17. Gradient of L over W^g:

By the definition W^g and (20), we have

$$\begin{aligned}\frac{\partial L}{\partial W^g} &= \sum_{j=1}^{T} \frac{\partial l(j)}{\partial g(j)} \frac{\partial g(j)}{\partial W^g s(j-1)} \frac{\partial W^g s(j-1)}{\partial W^g} \\ &= \sum_{j=1}^{T} \frac{\partial l(j)}{\partial g(t)} \circ (1 + g(j)) \circ (1 - g(j)) \otimes s(j-1) \qquad (28)\end{aligned}$$

1.2.3 Update parameters

Based on the derived backpropogation through time in last section, each parameter in LSTM is updated by

$$Var(t+1) = Var(t) - lr \cdot \frac{\partial L}{\partial Var(t)} \qquad (29)$$

where lr is learning rate.

1.3 Gated Recurrent Units (GRU)

GRU, shown in Figure 3, has a similar structure to LSTM, except that there is no gate responsible for accepting/rejecting input data. There is no consensus on whether GRU or LSTM is superior to the other. Since the procedure for forward propagation and backpropogation through time is similar to LSTM, we just listed the formulas without full derivation.

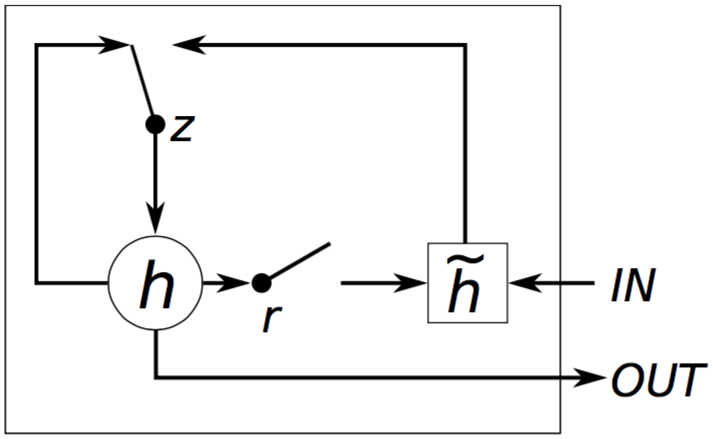

Figure 3: GRU structure

1.3.1 Forward Propagation

The forward propagation in GRU is shown as the belows:

$$z(t) = \sigma(U^z x(t) + W^z s(t-1)) \tag{30}$$
$$r(t) = \sigma(U^r x(t) + W^r s(t-1)) \tag{31}$$
$$h(t) = \tanh(U^h x(t) + W^h s(t-1) \circ z(t)) \tag{32}$$
$$s(t) = (1 - z(t)) \circ h(t) + z(t) \circ s(t-1) \tag{33}$$
$$o(t) = \text{softmax}(V s(t)) \tag{34}$$

1.3.2 Backpropogation Through Time

The gradients of cross entropy loss function over each variable is shown in the belows:

$$q(t) = U^h x(t) + W^h (s(t-1) \circ r(t)) \tag{35}$$

$$\delta(t) = \frac{\partial L}{\partial q(t)} = \sum_{j=1}^{T} \frac{\partial l(j)}{\partial q(t)} \tag{36}$$

$$\frac{\partial L}{\partial V} = \sum_{j=1}^{T} (\hat{y}(j) - y(j)) \otimes s(j) \tag{37}$$

$$\frac{\partial L}{\partial W^h} = \delta(t) \otimes (s(t-1) \circ r(t)) \tag{38}$$

$$\frac{\partial L}{\partial U^h} = x(t-1) \otimes \delta_t \tag{39}$$

$$\frac{\partial L}{\partial r} = W_r^T \cdot \delta_t \circ s(t-1) \tag{40}$$

$$\frac{\partial L}{\partial W^r} = (\frac{\partial L}{\partial r} \circ r(t) \circ (1 - r(t))) \otimes (s(t-1)) \tag{41}$$

$$\frac{\partial L}{\partial U^r} = (\frac{\partial L}{\partial r} \circ r(t) \circ (1 - r(t))) \otimes (x(t)) \tag{42}$$

$$\frac{\partial L}{\partial z(t)} = W_z^T \cdot \delta_t \circ r(t) \circ (s(t-2) - h(t-1)) \tag{43}$$

$$\frac{\partial L}{\partial W^z} = (\frac{\partial L}{\partial z(t)} \circ z(t) \circ (1 - z(t))) \otimes (s(t-1)) \tag{44}$$

$$\frac{\partial L}{\partial U^z} = (\frac{\partial L}{\partial z(t)} \circ z(t) \circ (1 - z(t))) \otimes (x(t)) \tag{45}$$

1.3.3 Update parameters

Based on the derived backpropogation through time in last section, each variable in GRU is updated by

$$Var(t+1) = Var(t) - lr \cdot \frac{\partial L}{\partial Var(t)} \tag{46}$$

where lr is learning rate.

2 Language Model on LSTM, and GRU

We have derived LSTM,and GRU. In this section, we will show how to employ LSTM on language model. As discussed in [4], in general we can use the numerical vector for each word in sentence as input, the label should be the numerical vector of next word in this sentence. Then LSTM, GRU networks can be trained on a bunch of sentences.

2.1 Data preprocessing

We select a 10000 English sentence dataset **europarl-eng-10000** from Prof. Koeh's [6]. Then a dictionary is built based on this dataset based on word frequency with vocabulary size = 2000. Infrequent words are replaced by "unk" or "UNKNOWNTOKEN". Each word is mapped to one unique index in this dictionary, and also represented by a unique vector $v(w) \in R^{2000}$. For example, if one word w's index is 2 in the dictionary, then

$$v(w) = (0, 0, 1, \underbrace{0, \cdots, 0}_{1997})^T$$

2.2 Emperimental Setting

We set learning rate as $lr = 0.01$ for LSTM, and 0.003 for GRU, maximum training epoches as 10.

2.3 Numerical Results

After data preprocessing, we train LSTM, and GRU on **europarl-eng-10000**, the loss value versus training epoch is shown in Figure 4. We can see loss values of both LSTM and GRU

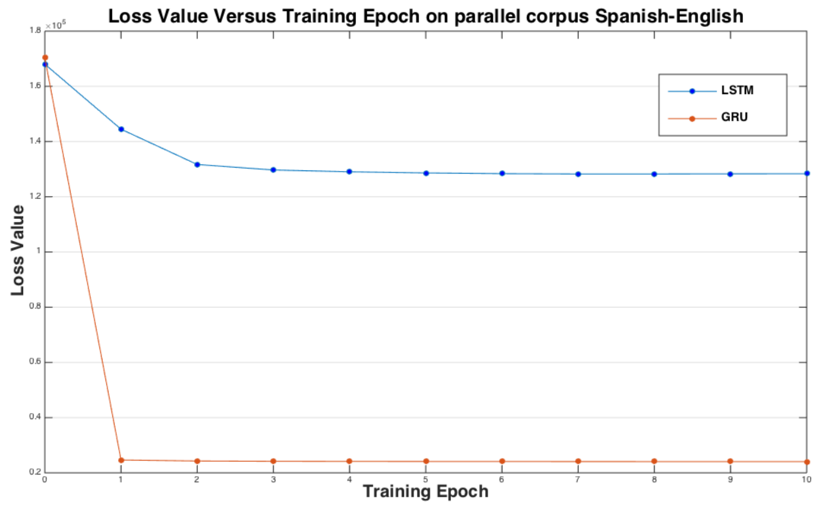

Figure 4: LSTM Loss Value during Training Iterations

are monotonously decreasing with the increase of training epoch. It implies our derivation and implementation of backpropogation through time on LSTM and GRU works. GRU can reach lower loss value, which may be caused by different gate structure, learning rate, and no bias in implemented LSTM(bias terms contained in implemented GRU).

3 GRU on Reranker

We also employ RNNs on reranker homework. Since in the last section, GRU can reach lower loss value, we select GRU instead of LSTM on reranker. The chosen training dataset is **train.ref**, and preprocessed in the same way shown in last section. After training GRU on **train.ref**, test dataset **train.100best** is used to generate the best hypothesis for each sentence. Since each sentence has 100 hypothesises, we calculate the loss value for each hypothesis by one time forward propagation. The hypothesis with lowest loss value is outputted as the best hypothesis for current sentence. To evaluate the quality of GRU reranker, **compute-bleu** script is used to calculate the BLEU score of produced best hypothesises. Furthermore, BLEU score of GRU reranker is compared with the given **rerank** script. Since given **rerank** script considers language model, translation model, lexical translation model, and GRU reranker is built on only language model, for fair comparison, we set the weights for translation model, and lexical translation model as 0.0.

3.1 Experimental Setting

The maximum training epoch is set as 20. Learning rate is set as 0.003. Vocabulary size is 3000.

3.2 Numerical Results

The BLEU scores for GRU reranker and modified given **reranker** script is shown in Table 1.

Table 1: Bleu Scores

Method	BLEU score
GRU reranker	0.24620
given rerank	0.24711

As shown in Table 1, we can see the BLEU score of GRU reranker is very close to that of modified given **rerank**. Since GRU reranker is built on language model, it seems that GRU reranker may be competitive to traditional language model by n-gram counting.

4 Conclusion

In this project, we derive and implement LSTM, GRU language model. The numerical results show GRU can reach lower loss value. GRU is further employed on reranker homework, and compared with given **reranker** in pure language model version. The BLEU scores of them are very close, which implies that GRU language model may be competitive to traditional language model by word counting.

References

[1] Philipp Koehn, Franz Josef Och, Daniel Marcu. *Statistical Phrase-Based Translation*, 2003

[2] Sepp Hochreiter, Jurgen Schmidhuber, *Long Short-Term Memory*, Neural Computation 9(8):1735 1780, 1997

[3] Y. Benigio, P. Simard, P. Frasconi, *Learning Long-Term Dependence with Gradient Descent is Difficult*, IEEE Transaction on Neural Network, Vol 5, No. 2, 1994

[4] wildML,*http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-2-implementing-a-language-model-rnn-with-python-numpy-and-theano/*

[5] Chung, Junyoung, et al. *?Empirical evaluation of gated recurrent neural networks on sequence modeling.? (2014)*

[6] Philipp Koehn, *http://www.statmt.org/europarl/*