深度学习技术与应用（9）

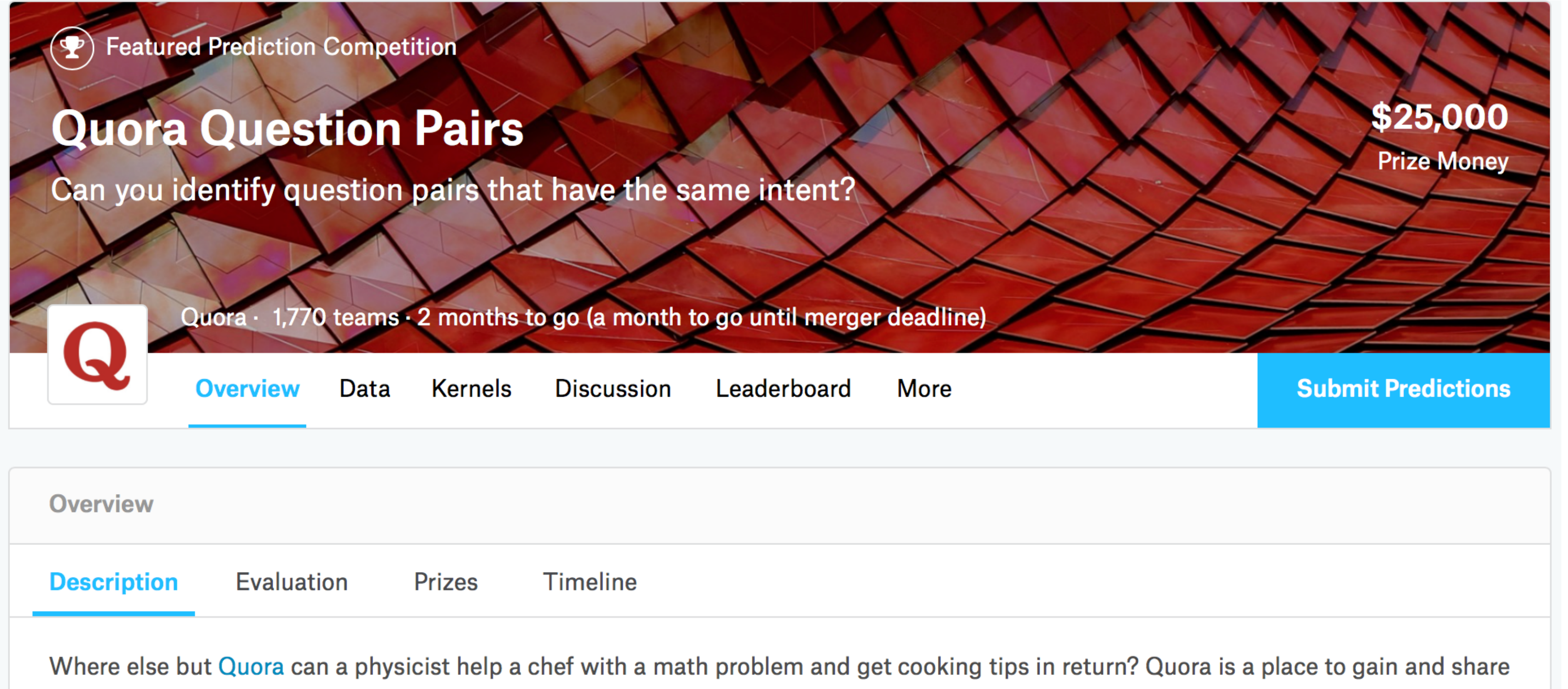
Deep Learning: Techniques and Applications (9)

Ge Li

Peking University

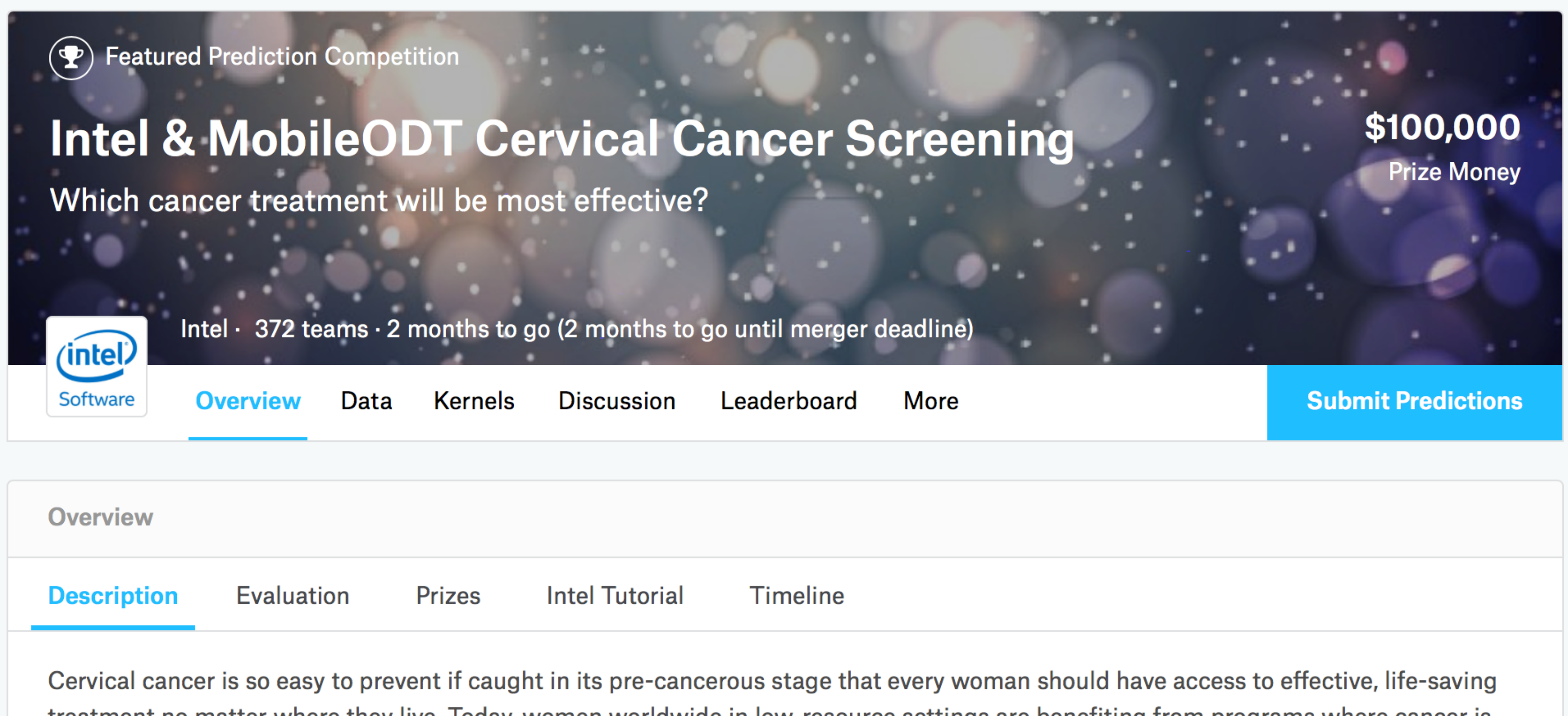
Project 1

- https://www.kaggle.com/c/quora-question-pairs



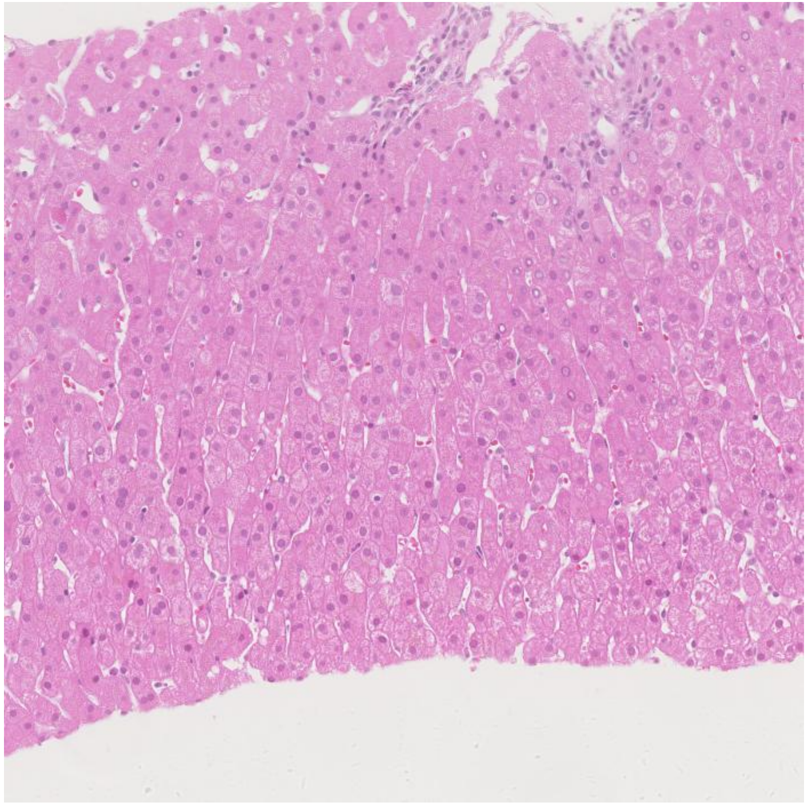
Project 2

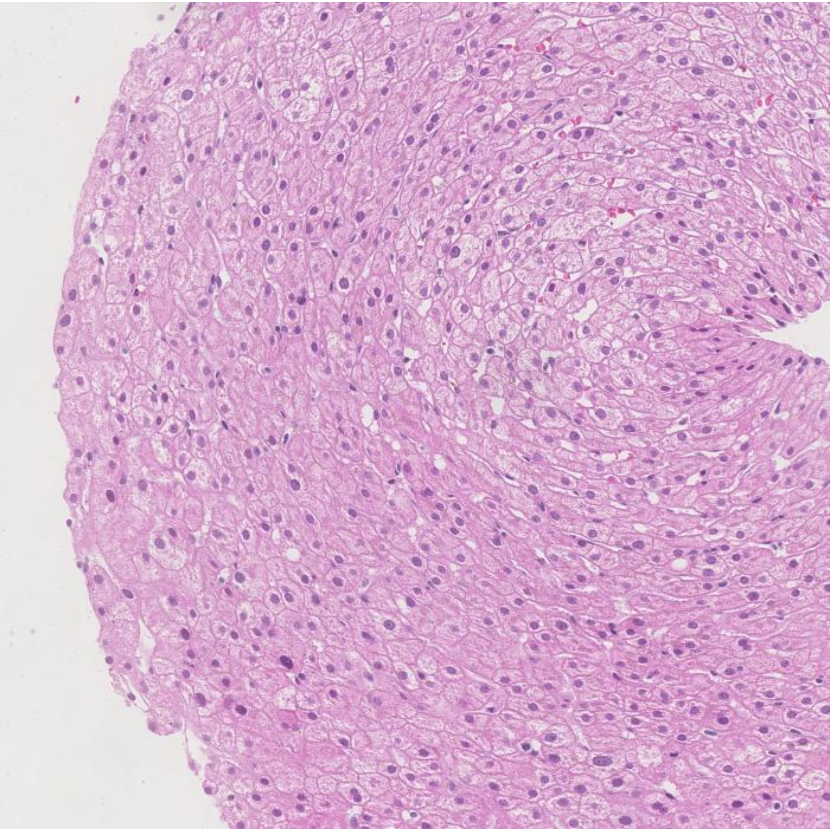
- https://www.kaggle.com/c/intel-mobileodt-cervical-cancer-screening



Project 3

- 肝病分期预测





Project 4

■ 对话生成引擎

Hi.
Looks like things worked out tonight, huh?

You know Chastity?
I believe we share an art instructor

Have fun tonight?
Tons

I looked for you back at the party, but you always seemed to be "occupied".
I was?
You never wanted to go out with 'me, did you?

Well, no...
Then that's all you had to say.
But
You always been this selfish?

Then Guillermo says, "If you go any lighter, you're gonna look like an extra on 90210."
No...

do you listen to this crap?
What crap?
Me. This endless ...blonde babble. I'm like, boring myself.
Thank God! If I had to hear one more story about your coiffure...

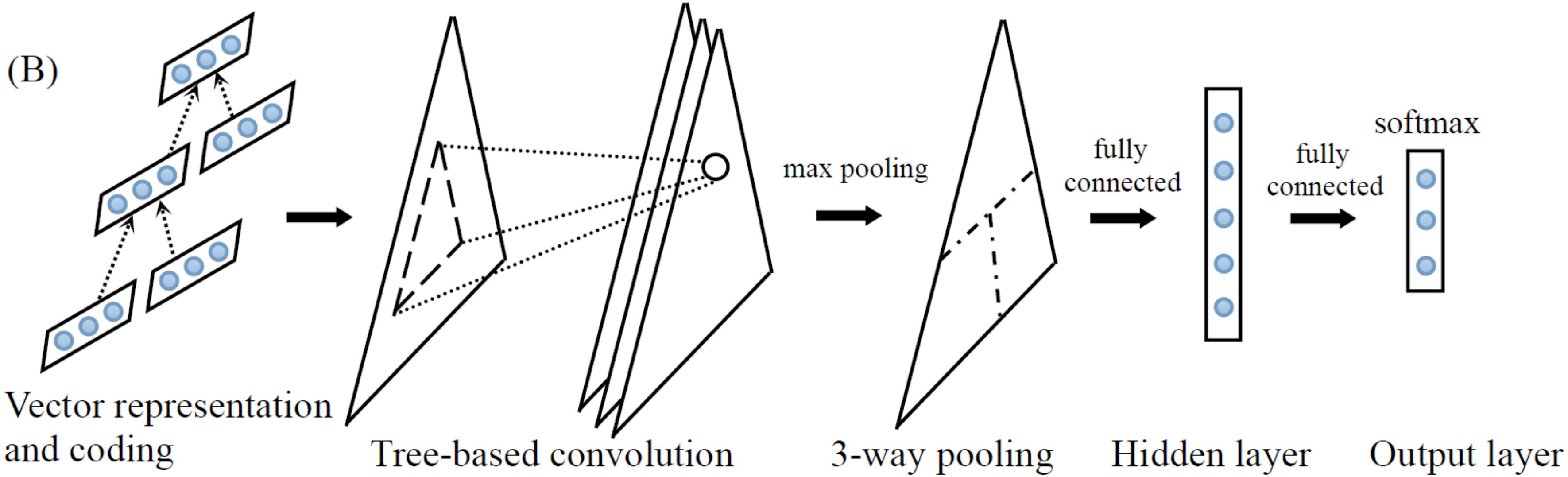
知道习近平是谁吗？	你的话题涉及敏感词汇哦，我们聊点别的吧。
知道什么叫什么玩意吗	要讲文明用语哦
知道了	好滴，还有什么可以帮到您的？
吱一声啊	吱……
只要等一小会	好的，随时恭候。
真是太谢谢你了	为您服务是我的荣幸哦
真傻！	您这么说我会伤心的
真烂	您这么说我会伤心的
真乖	妈妈说要懂礼貌的哦
真够猪的	要讲文明用语哦
真的呀	是啊
真的太麻烦你了	为您服务是我的荣幸哦
真的没话说	我们可以聊点其他的，比如天气？
真的吗	我很真诚的
真的假的	客服机器人是很真诚的，不要怀疑我的真心哦。
真的好喜欢你	我也是哦
真的不喜欢你	您这么说我会伤心的
真不好玩	我才出生没多久呢，好多不懂的您多宽容点呗
真不聪明	我才出生没多久呢，好多不懂的您多宽容点呗
这样挺好	恩，我也觉得呢
这样啊	是啊
这是一个什么样的产品	我是智能客服机器人，我可以回答您的问题，陪您聊天哦
这是我的问题，我道歉	没关系啦
这是什么啊	亲，请直接输入您想问的关键词，我会立即为您搜索答案的。
这什么呀	亲，请直接输入您想问的关键词，我会立即为您搜索答案的。
这么客气干什么	妈妈说要懂礼貌的哦
这么客气啊。	妈妈说要懂礼貌的哦

Project 5

```c
int main()
{
        int f(int a,int min);
        int n,a[150],i,b;
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                scanf("%d",&a[i]);
        }
        for(i=0;i<n;i++)
        {
                b=f(a[i],2);
                printf("%d\n",b);
        }
        return 0;
}


int f(int a,int min)
{
        int result=1,i;
        if(a < min)
        {
        return 0;
        }
        for(i = min;i<a;i++)
        {
                if(a % i == 0)
                {
                result += f(a/i,i);
                }
        }
        return result;
}
```

```c

void initarray ( int a[] ,int b , int length ); // ?????
void base ( int a[maxlen] ); // ??
int twice ( int a[maxlen]    ); // ????
int main ( void ) // ???
{
        int a[maxlen];
        int t[maxlen];
        int i,j,k,l;
        i = 0;
        for ( i = 0 ; i < maxlen ; i ++ )
        {
                a[i] = -2;
        }
        i = 0;
        k = -2;
        while( a[i] != -1 )
        {
                while ( k != 0 )
                {
                        cin>>k;
                        a[i] = k ;
                        i ++ ;
                }
                base ( a );
                j = twice ( a );
                cout<<j<<endl;
                initarray ( a , -2 ,maxlen);
                i = 0 ;
                k = -2 ;
        }
        return 1;
}

void initarray (int a[] ,int b , int length)
{
        int i;
        for ( i = 0 ; i < length ; i++)
        {
                a[i] = b;
        }
}

void base ( int a[maxlen] )
{
        int b[maxinputint+1];
        int i,j;
        initarray(b,0,maxinputint);
        for ( i = 0 ; i<maxlen; i ++ )
        {
                if ( a[i] > 0 )
                {
                        b[a[i]]++;
                }
        }
        j = 0;
        for ( i = 0 ; i<= maxinputint; i ++ )
        {
                if ( b[i] == 1 )
                {
                        a[j] = i;
                        j++;
                }
        }
}

int twice ( int a[maxlen] )
{
int i,j,t,result;

i = 0;
j = 0;
t = 0;
result = 0;
while ( ( a[i] <= 50 ) && ( a[j] > 0 ))
{
        t = a[i] * 2;
        j = i ;
        while ( (a[j] <= t) && ( a[j] > 0 ) )
        {
                if ( a[j] == t )
                {
                        result ++ ;
                };
                j++ ;
        }
        j = 0;
        i ++ ;
}
return result;
}
```

Tree-based Convolutional Neural Network



Tree-Based Convolutional Neural Network for Programming Language Processing

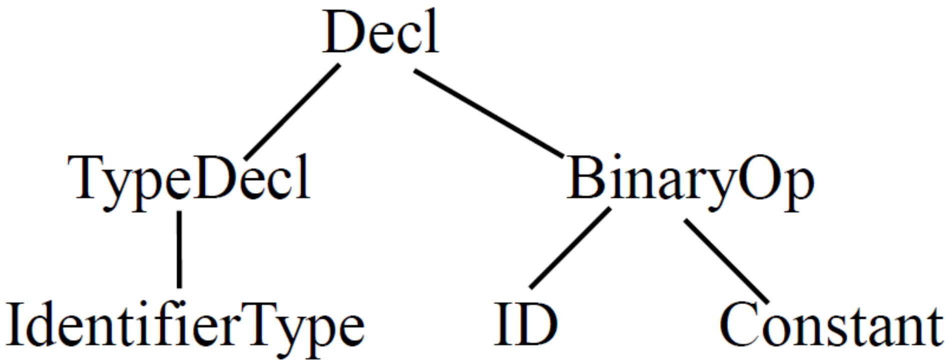


Figure 2: The AST corresponding to the C code snippet "int a=b+3;" A salient difference of ASTs and natural language syntax trees is that the non-leaf nodes in the latter do not have explicit meaning (e.g. a noun phrase); but AST nodes do (e.g. BinaryOp).

Tree-Based Convolutional Neural Network for Programming Language Processing

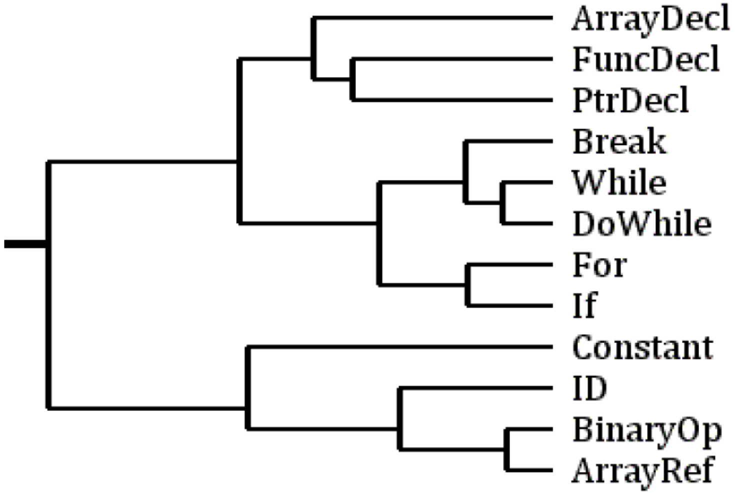


Figure 3: Hierarchical clustering results based on vector representations for AST nodes.

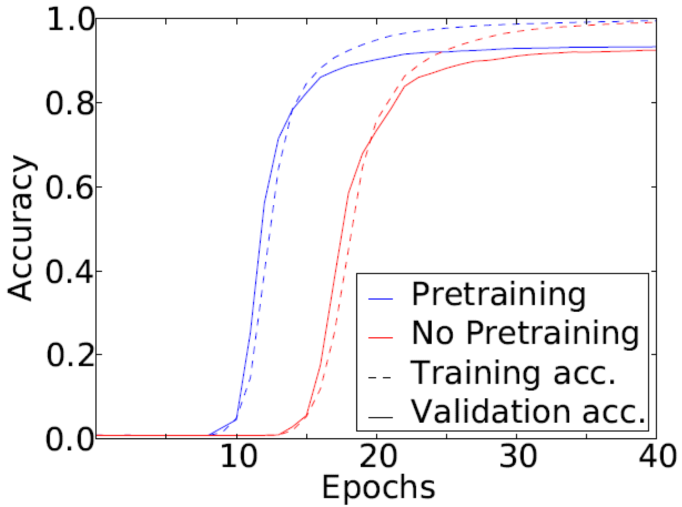


Figure 4: Learning curves with and without pretraining.

Tree-Based Convolutional Neural Network for Programming Language Processing

Method	Test Acc.
linear SVM+BoW	52.0
RBF SVM+BoW	83.9
linear SVM+BoT	72.5
RBF SVM+BoT	88.2
RNN	84.8
TBCNN	**94.0**

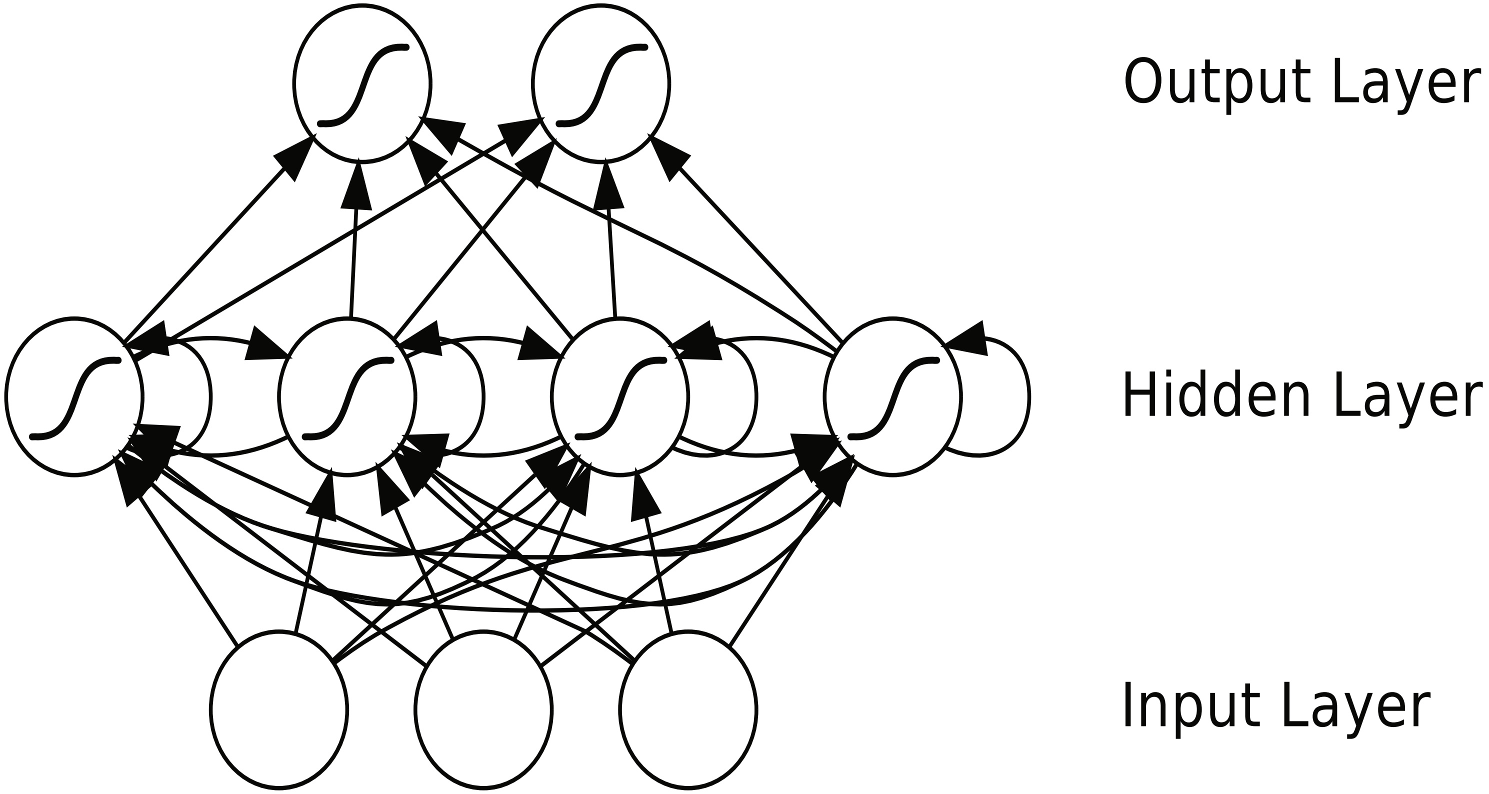
Table 1: The accuracy (in percentage) of 104-label program classifications.

Tree-Based Convolutional Neural Network for Programming Language Processing

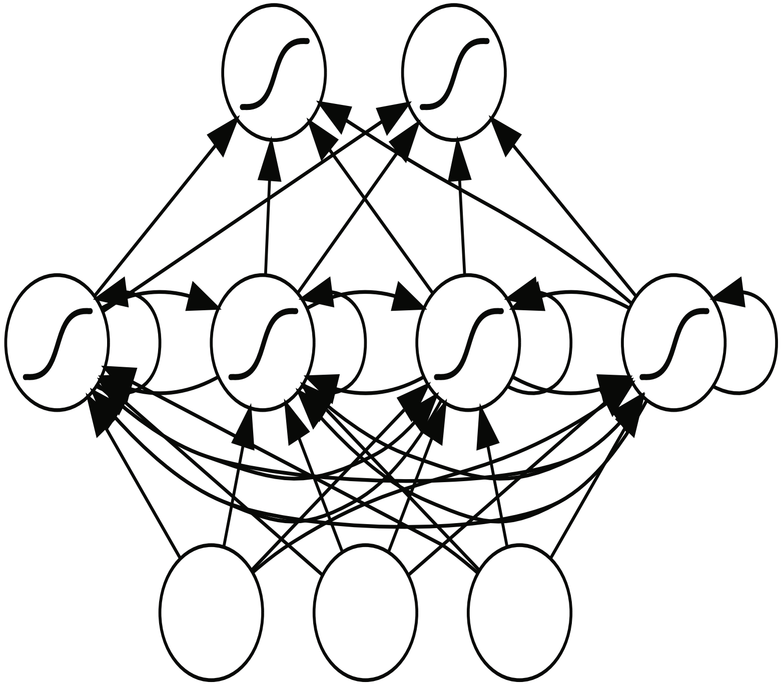
Classifier	Features	Accuracy %
Random/majority	–	50.0
RBF SVM	Bag-of-word	62.3
RBF SVM	Bag-of-tree	77.1
TBCNN	Learned	**89.1**

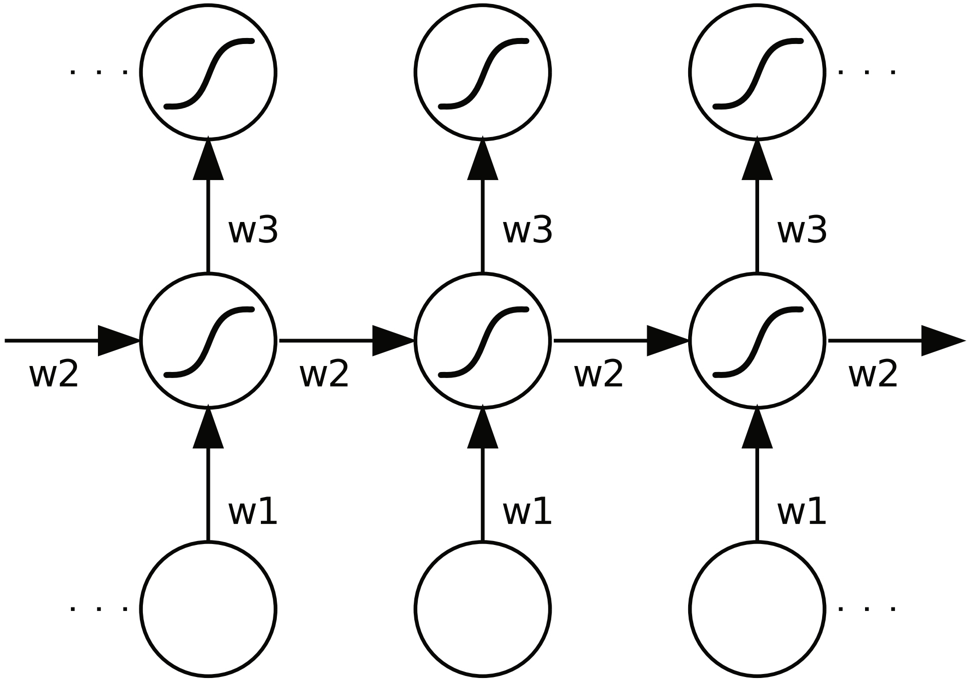
Table 2: Accuracy of detecting bubble sort.

RNN

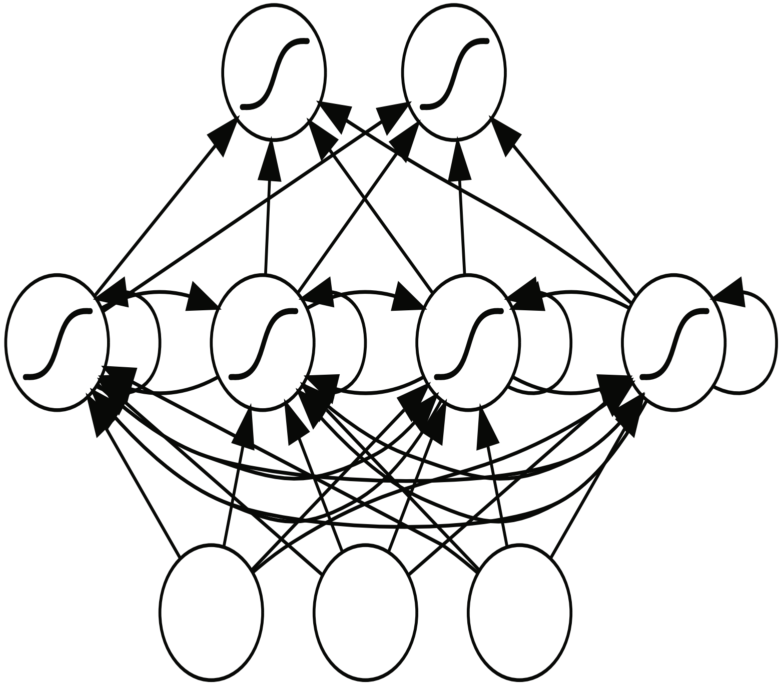


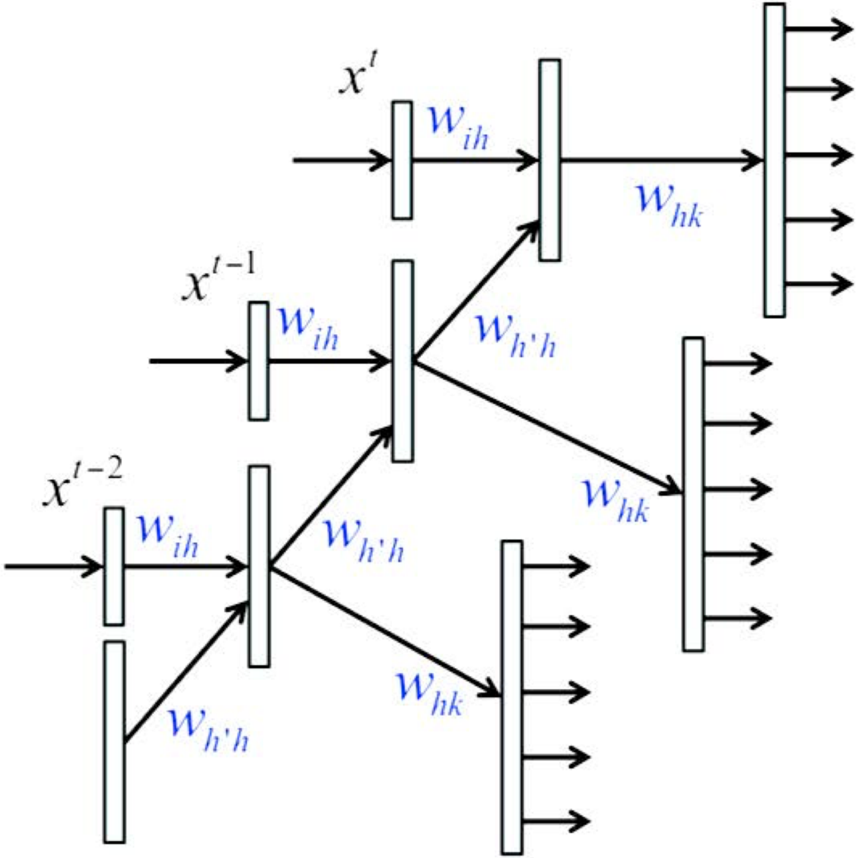
RNN



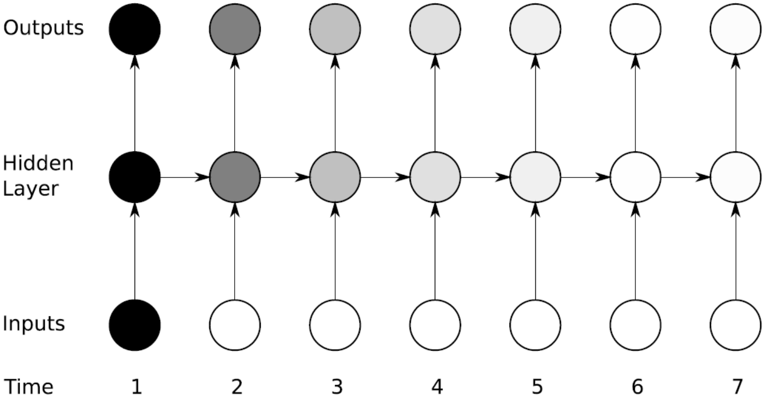


RNN

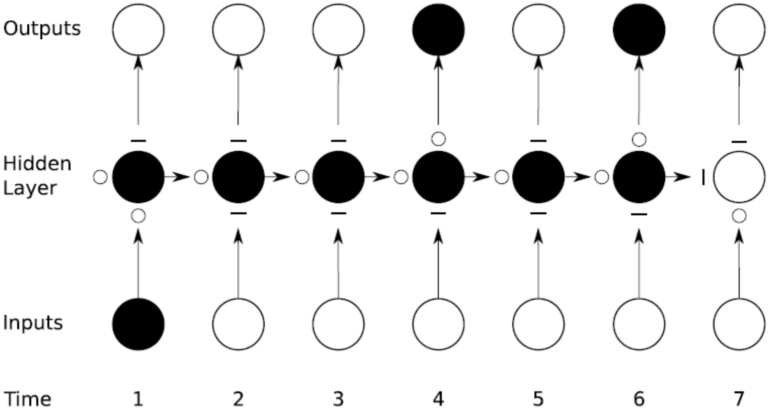




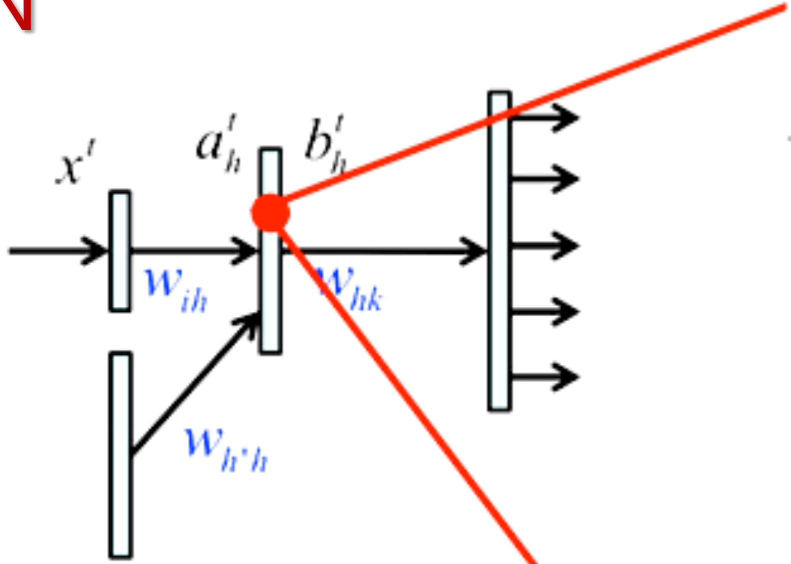
LSTM 的提出

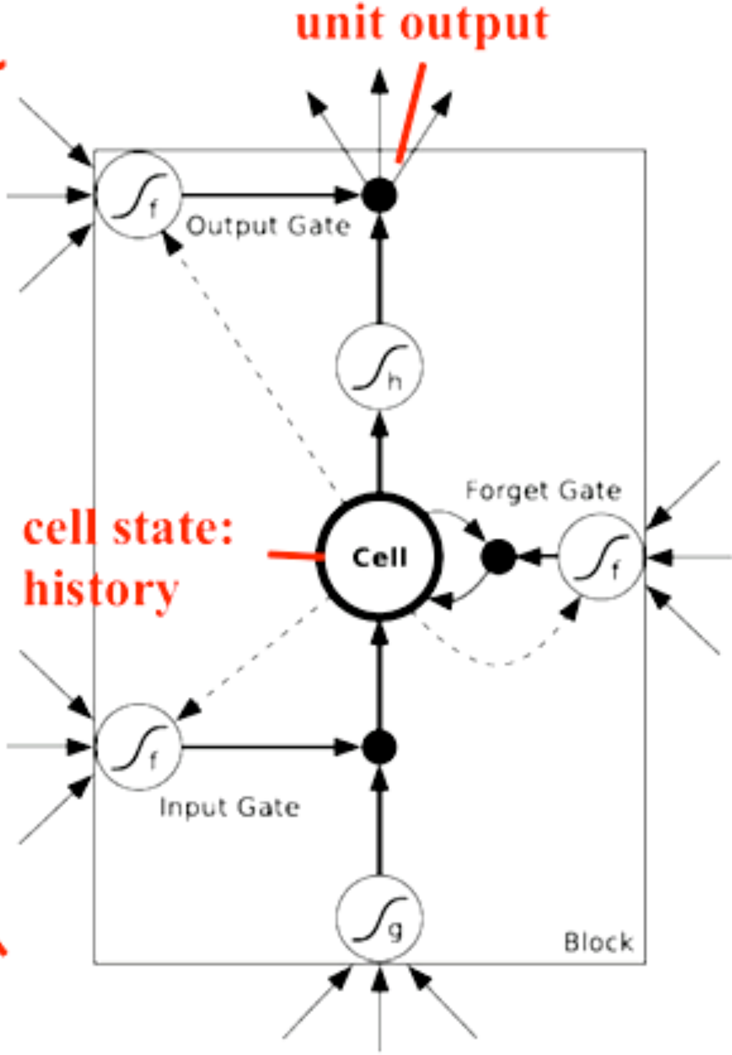


LSTM 的提出



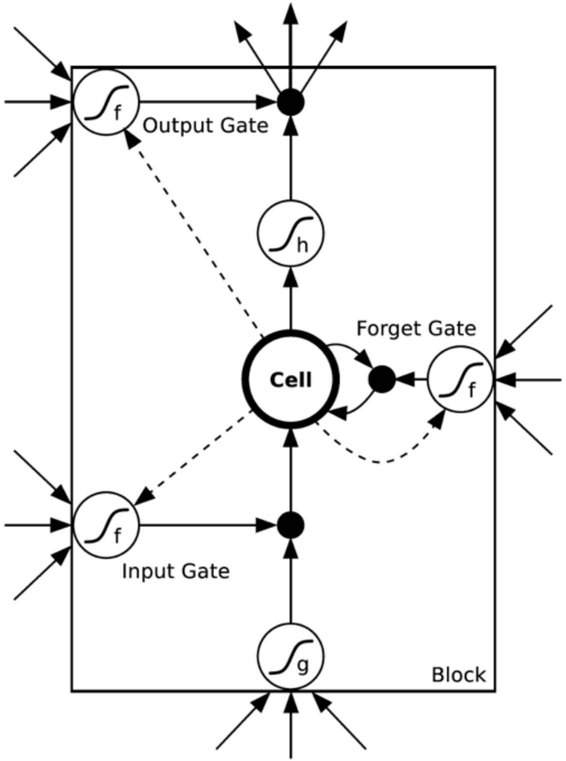
LSTM RNN

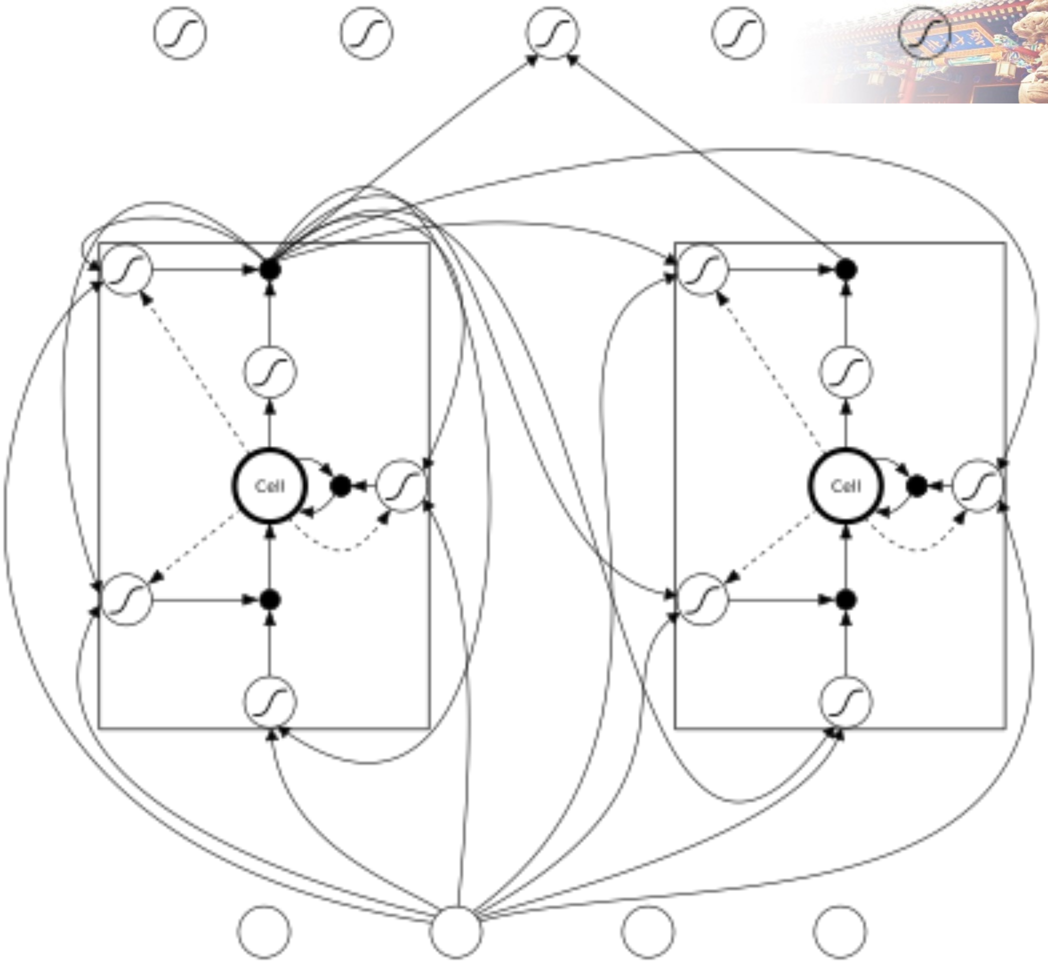




$$\left.\begin{array}{l} \iota - \text{Input Gate} \\ \phi - \text{Forget Gata} \\ \omega - \text{Output Gate} \end{array}\right\} \in (0,1)$$

LSTM RNN





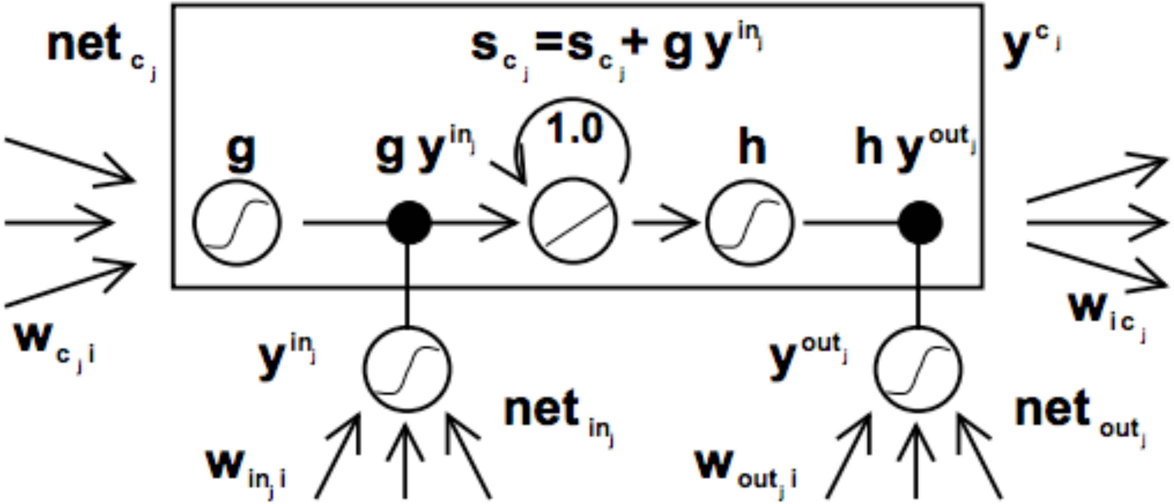
最早的LSTM

Long short-term memory

S Hochreiter, J Schmidhuber - Neural computation, 1997 - MIT Press

Learning to store information over extended time intervals by recurrent backpropagation takes a very long time, mostly because of insufficient, decaying error backflow. We briefly review Hochreiter's (1991) analysis of this problem, then address it by introducing a novel,

被引用次数：4126　相关文章　所有 40 个版本　引用　保存　更多



LSTM Inventor

Deep learning in neural networks: An overview

J Schmidhuber - Neural networks, 2015 - Elsevier

Abstract In recent years, deep artificial neural networks (including recurrent ones) have won numerous contests in pattern recognition and machine learning. This historical survey compactly summarizes relevant work, much of it from the previous millennium. Shallow and Deep Learners are distinguished by the depth of their credit assignment paths, which are chains of possibly learnable, causal links between actions and effects. I review deep ...

Long short-term memory S Hochreiter, J Schmidhuber Neural computation 9 (8), 1735-1780	4135	1997
Deep learning in neural networks: An overview J Schmidhuber Neural networks 61, 85-117	1228	2015
Multi-column deep neural networks for image classification D Ciregan, U Meier, J Schmidhuber Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on ...	1126	2012
A novel connectionist system for unconstrained handwriting recognition A Graves, M Liwicki, S Fernández, R Bertolami, H Bunke, J Schmidhuber IEEE transactions on pattern analysis and machine intelligence 31 (5), 855-868	586	2009
Learning to forget: Continual prediction with LSTM FA Gers, J Schmidhuber, F Cummins Neural computation 12 (10), 2451-2471	486	2000

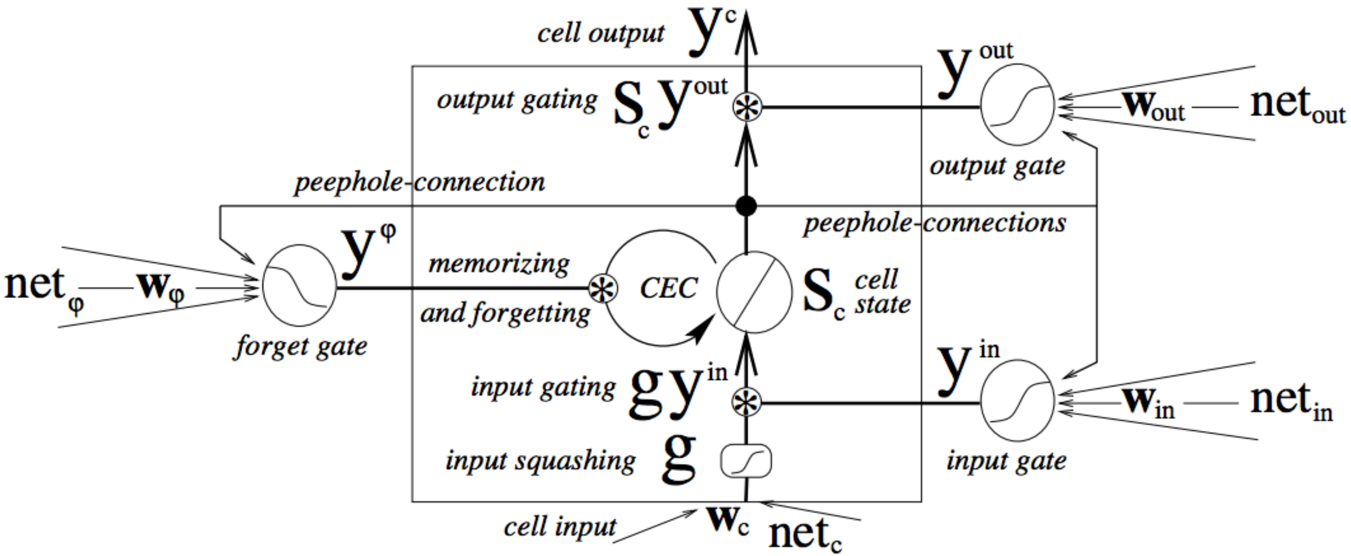
常用的LSTM

Recurrent nets that time and count

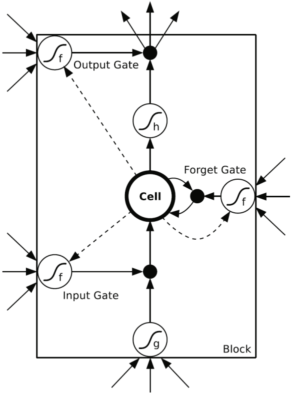
FA Gers, J Schmidhuber - … 2000, Proceedings of the IEEE-INNS …, 2000 - ieeexplore.ieee.org

Abstract: The size of the time intervals between events conveys information essential for numerous sequential tasks such as motor control and rhythm detection. While hidden Markov models tend to ignore this information, recurrent neural networks (RNN) can in

被引用次数：87　相关文章　所有 7 个版本　引用　保存



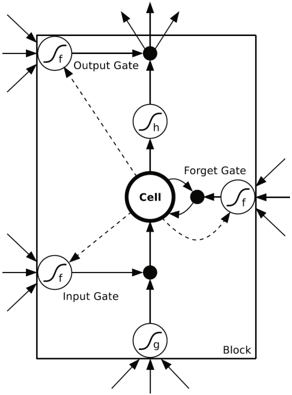
LSTM 单元



- 设 LSTM 隐藏层共包含 H 个神经元，下标 h 表示其中之一；
- 设 LSTM 隐藏层共包含 C 个 Cell，下标 c 表示某个 Cell；
- 当前的 LSTM 单元中 Input Gate,Forget Gate, Output Gate 分别用下标 α, β, γ 标识；
- LSTM 单元在 t 时刻的输入：z_h^t， t 时刻的输出：a_h^t；

  对于仅包含一个 Cell 的 LSTM 单元，$z_h^t = z_c^t, a_c^t = a_h^t$；

LSTM 单元



LSTM 单元的输入:

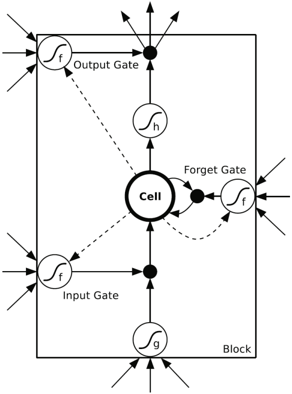
$$z_h^t = \sum_{i=1}^{I} w_{ci} x_i^t + \sum_{h=1}^{H} w_{ch} a_h^{t-1}$$

注意：a_h 表示来自于其他 LSTM 单元的输出 a_c；

LSTM 单元的输出:

$$a_h^t = a_c^t$$

LSTM 单元



Input Gate:

$$z_\alpha^t = \sum_{i=1}^{I} w_{\alpha i} x_i^t + \sum_{h=1}^{H} w_{\alpha h} a_h^{t-1} + \sum_{c=1}^{C} w_{\alpha c} s_c^{t-1}$$

$$a_\alpha^t = f(z_\alpha^t)$$

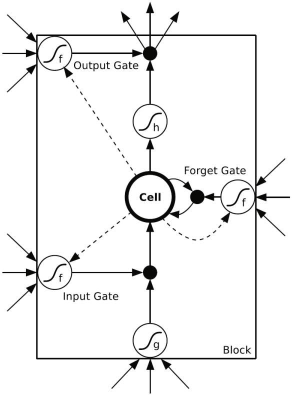
注意：a_h 表示来自于其他 LSTM 单元的输出 a_c；

Forget Gate:

$$z_\beta^t = \sum_{i=1}^{I} w_{\beta i} x_i^t + \sum_{h=1}^{H} w_{\beta h} a_h^{t-1} + \sum_{c=1}^{C} w_{\beta c} s_c^{t-1}$$

$$a_\beta^t = f(z_\beta^t)$$

LSTM 单元



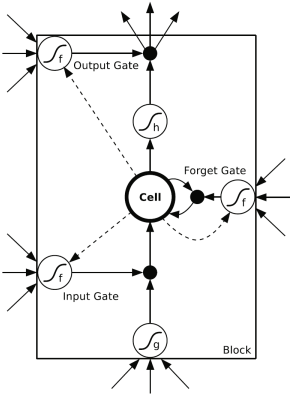
Cells:

$$z_c^t = \sum_{i=1}^{I} w_{ci} x_i^t + \sum_{h=1}^{H} w_{ch} a_h^{t-1}$$

注意：a_h 表示来自于其他 LSTM 单元的输出 a_c；

$$s_c^t = a_\alpha^t g(z_c^t) + a_\beta^t s_c^{t-1}$$

LSTM 单元



Output Gate:

$$z_\gamma^t = \sum_{i=1}^{I} w_{\gamma i} x_i^t + \sum_{h=1}^{H} w_{\gamma h} a_h^{t-1} + \sum_{c=1}^{C} w_{\gamma c} s_c^{t-1}$$

$$a_\gamma^t = f(z_\gamma^t)$$

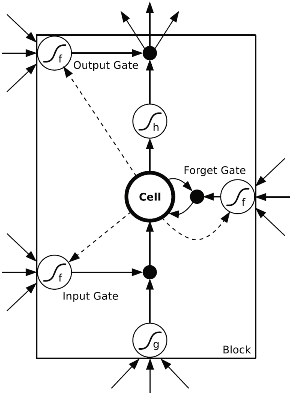
Cell Outputs:

$$a_c^t = a_\gamma^t h(s_c^t)$$

LSTM Outputs:

$$z_k^t = \sum_{k=1}^{K} w_{kc} a_h^t$$

LSTM 单元



统计要计算的参数：

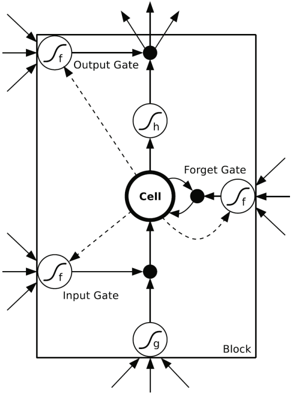
$$w_{\alpha i}, w_{\alpha h}, w_{\alpha c}$$

$$w_{\beta i}, w_{\beta h}, w_{\beta c}$$

$$w_{\gamma i}, w_{\gamma h}, w_{\gamma c}$$

$$w_{ci}, w_{ch}, w_{kc}$$

LSTM 单元



$$w_{\alpha i}: \frac{\partial J(W,b)}{\partial w_{\alpha i}} = \frac{\partial J(W,b)}{\partial z_\alpha^t}\frac{\partial z_\alpha^t}{\partial w_{\alpha i}} = \frac{\partial J(W,b)}{\partial z_\alpha^t} x_i^t$$

$$w_{\alpha h}: \frac{\partial J(W,b)}{\partial w_{\alpha h}} = \frac{\partial J(W,b)}{\partial z_\alpha^t}\frac{\partial z_\alpha^t}{\partial w_{\alpha h}} = \frac{\partial J(W,b)}{\partial z_\alpha^t} a_h^{t-1}$$

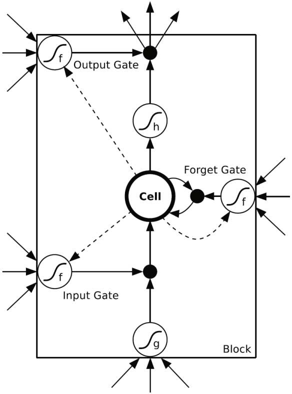
$$w_{\alpha c}: \frac{\partial J(W,b)}{\partial w_{\alpha c}} = \frac{\partial J(W,b)}{\partial z_\alpha^t}\frac{\partial z_\alpha^t}{\partial w_{\alpha c}} = \frac{\partial J(W,b)}{\partial z_\alpha^t} s_c^{t-1}$$

$$w_{\beta i}: \frac{\partial J(W,b)}{\partial w_{\beta i}} = \frac{\partial J(W,b)}{\partial z_\beta^t}\frac{\partial z_\beta^t}{\partial w_{\beta i}} = \frac{\partial J(W,b)}{\partial z_\beta^t} x_i^t$$

$$w_{\beta h}: \frac{\partial J(W,b)}{\partial w_{\beta h}} = \frac{\partial J(W,b)}{\partial z_\beta^t}\frac{\partial z_\beta^t}{\partial w_{\beta h}} = \frac{\partial J(W,b)}{\partial z_\beta^t} a_h^{t-1}$$

$$w_{\beta c}: \frac{\partial J(W,b)}{\partial w_{\beta c}} = \frac{\partial J(W,b)}{\partial z_\beta^t}\frac{\partial z_\beta^t}{\partial w_{\beta c}} = \frac{\partial J(W,b)}{\partial z_\beta^t} s_c^{t-1}$$

LSTM 单元



$$w_{\gamma i}: \frac{\partial J(W,b)}{\partial w_{\gamma i}} = \frac{\partial J(W,b)}{\partial z_\gamma^t}\frac{\partial z_\gamma^t}{\partial w_{\gamma i}} = \frac{\partial J(W,b)}{\partial z_\gamma^t} x_i^t$$

$$w_{\gamma h}: \frac{\partial J(W,b)}{\partial w_{\gamma h}} = \frac{\partial J(W,b)}{\partial z_\gamma^t}\frac{\partial z_\gamma^t}{\partial w_{\gamma h}} = \frac{\partial J(W,b)}{\partial z_\gamma^t} a_h^{t-1}$$

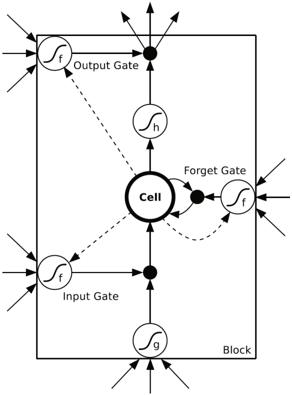
$$w_{\gamma c}: \frac{\partial J(W,b)}{\partial w_{\gamma c}} = \frac{\partial J(W,b)}{\partial z_\gamma^t}\frac{\partial z_\gamma^t}{\partial w_{\gamma c}} = \frac{\partial J(W,b)}{\partial z_\gamma^t} s_c^{t-1}$$

$$w_{ci}: \frac{\partial J(W,b)}{\partial w_{ci}} = \frac{\partial J(W,b)}{\partial z_h^t}\frac{\partial z_h^t}{\partial w_{ci}} = \frac{\partial J(W,b)}{\partial z_c^t} x_i^t$$

$$w_{ch}: \frac{\partial J(W,b)}{\partial w_{ch}} = \frac{\partial J(W,b)}{\partial z_h^t}\frac{\partial z_h^t}{\partial w_{ch}} = \frac{\partial J(W,b)}{\partial z_h^t} a_h^{t-1}$$

$$w_{kc}: \frac{\partial J(W,b)}{\partial w_{kc}} = \frac{\partial J(W,b)}{\partial z_k^t}\frac{\partial z_k^t}{\partial w_{kc}} = \frac{\partial J(W,b)}{\partial z_k^t} a_h^t$$

LSTM 单元



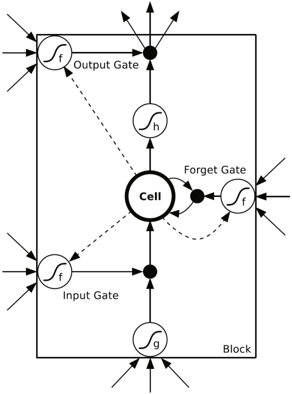
统计要计算的梯度：

$$\frac{\partial J(W,b)}{\partial z_\alpha^t} \quad \frac{\partial J(W,b)}{\partial z_\beta^t} \quad \frac{\partial J(W,b)}{\partial z_\gamma^t}$$

$$\frac{\partial J(W,b)}{\partial z_c^t} \quad \frac{\partial J(W,b)}{\partial z_h^t} \quad \frac{\partial J(W,b)}{\partial z_k^t}$$

LSTM 单元

梯度计算之一：

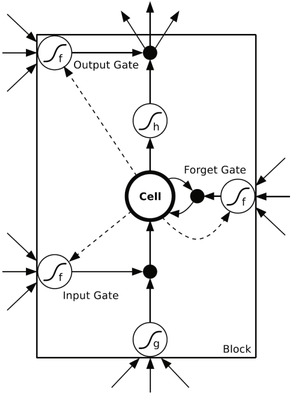


若 t=T，则：
$$\frac{\partial J(W,b)}{\partial z_k^T} = \frac{\partial J(W,b)}{\partial a_k^T}\frac{\partial a_k^T}{\partial z_k^T} = \frac{\partial J(W,b)}{\partial a_k^T} output'(\cdot)$$

否则：
$$\frac{\partial J(W,b)}{\partial z_k^t} = \frac{\partial J(W,b)}{\partial a_k^t}\frac{\partial a_k^t}{\partial z_k^t} = \sum_h^H \frac{\partial J(W,b)}{\partial z_h^{t+1}}\frac{\partial a_k^t}{\partial z_k^t} = \sum_h^H \frac{\partial J(W,b)}{\partial z_h^{t+1}} output'(\cdot)$$

LSTM 单元

梯度计算之二：



$$\frac{\partial J(W,b)}{\partial z_\gamma^t} = \sum_{c=1}^{C} \frac{\partial J(W,b)}{\partial a_c^t} \frac{\partial a_c^t}{\partial a_\gamma^t} \frac{\partial a_\gamma^t}{\partial z_\gamma^t}$$

因为：$a_c^t = a_\gamma^t h(s_c^t)$

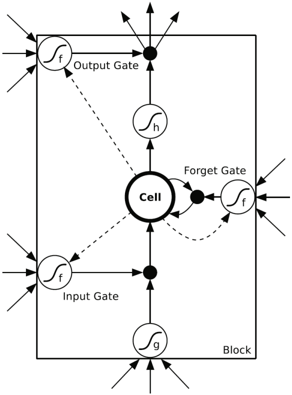
所以：

$$= \sum_{c=1}^{C} \frac{\partial J(W,b)}{\partial a_c^t} h(s_c^t) f'(z_\gamma^t)$$

$$= f'(z_\gamma^t) \sum_{c=1}^{C} \frac{\partial J(W,b)}{\partial a_c^t} h(s_c^t)$$

LSTM 单元

梯度计算之三：



$$\frac{\partial J(W,b)}{\partial a_c^t} = \sum_k^K \frac{\partial J(W,b)}{\partial z_k^t}\frac{\partial z_k^t}{\partial a_c^t} + \sum_h^H \frac{\partial J(W,b)}{\partial z_h^{t+1}}\frac{\partial z_h^{t+1}}{\partial a_c^t}$$

因为：

$$z_h^{t+1} = \sum_{i=1}^I w_{ci}x_i^{t+1} + \sum_{h=1}^H w_{ch}a_h^t$$

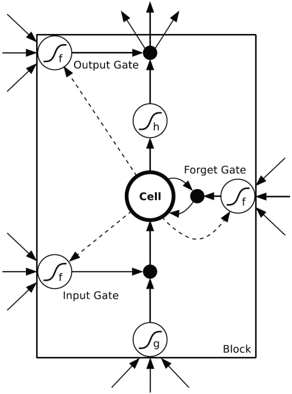
$$z_k^t = \sum_{k=1}^K w_{kc}a_h^t$$

所以：

$$\frac{\partial J(W,b)}{\partial a_c^t} = \sum_k^K \frac{\partial J(W,b)}{\partial z_k^t} w_{kc} + \sum_h^H \frac{\partial J(W,b)}{\partial z_h^{t+1}} w_{ch}$$

LSTM 单元

梯度计算之四：



$$\frac{\partial J(W,b)}{\partial z_h^t} = \sum_{c=1}^{C} \frac{\partial J(W,b)}{\partial s_c^t} \frac{\partial s_c^t}{\partial z_h^t}$$

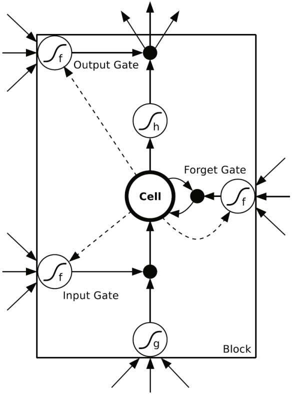
因为：$s_c^t = a_\alpha^t g(z_c^t) + a_\beta^t s_c^{t-1}$

所以：

$$\frac{\partial J(W,b)}{\partial z_h^t} = \sum_{c=1}^{C} \frac{\partial J(W,b)}{\partial s_c^t} a_\alpha^t g'(z_c^t)$$

LSTM 单元

梯度计算之五：



$$\frac{\partial J(W,b)}{\partial z_\beta^t} = \sum_{c=1}^{C} \frac{\partial J(W,b)}{\partial s_c^t} \frac{\partial s_c^t}{\partial z_\beta^t}$$

因为：$s_c^t = a_\alpha^t g(z_c^t) + a_\beta^t s_c^{t-1}$

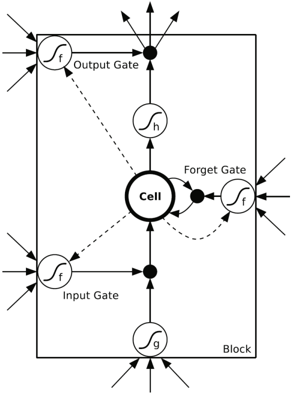
所以：

$$\frac{\partial J(W,b)}{\partial z_h^t} = \sum_{c=1}^{C} \frac{\partial J(W,b)}{\partial s_c^t} \frac{\partial a_\beta^t}{\partial z_\beta^t} s_c^{t-1}$$

$$= f'(z_\beta^t) \sum_{c=1}^{C} \frac{\partial J(W,b)}{\partial s_c^t} s_c^{t-1}$$

LSTM 单元

梯度计算之六：



$$\frac{\partial J(W,b)}{\partial z_\alpha^t} = \sum_{c=1}^{C} \frac{\partial J(W,b)}{\partial s_c^t} \frac{\partial s_c^t}{\partial z_\alpha^t}$$

因为：$s_c^t = a_\alpha^t g(z_c^t) + a_\beta^t s_c^{t-1}$

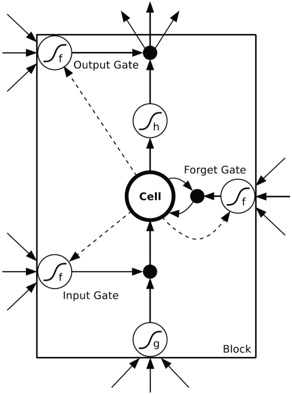
所以：

$$\frac{\partial J(W,b)}{\partial z_h^t} = \sum_{c=1}^{C} \frac{\partial J(W,b)}{\partial s_c^t} \frac{\partial a_\alpha^t}{\partial z_\alpha^t} g(z_c^t)$$

$$= f'(z_\alpha^t) \sum_{c=1}^{C} \frac{\partial J(W,b)}{\partial s_c^t} g(z_c^t)$$

LSTM 单元

梯度计算之七：



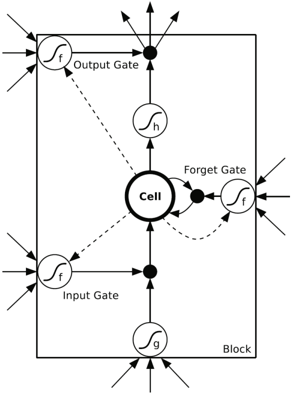
焦点集中在：$\dfrac{\partial J(W,b)}{\partial s_c^t}$

因为：
$$\frac{\partial J(W,b)}{\partial s_c^t} = \frac{\partial J(W,b)}{\partial a_c^t}\frac{\partial a_c^t}{\partial s_c^t} + \frac{\partial J(W,b)}{\partial s_c^{t+1}}\frac{\partial s_c^{t+1}}{\partial s_c^t} + \frac{\partial J(W,b)}{\partial z_\alpha^{t+1}}\frac{\partial z_\alpha^{t+1}}{\partial s_c^t} + \frac{\partial J(W,b)}{\partial z_\beta^{t+1}}\frac{\partial z_\beta^{t+1}}{\partial s_c^t} + \frac{\partial J(W,b)}{\partial z_\gamma^{t+1}}\frac{\partial z_\gamma^{t+1}}{\partial s_c^t}$$

所以：
$$\frac{\partial J(W,b)}{\partial s_c^t} = \frac{\partial J(W,b)}{\partial a_c^t} a_\gamma^t h'(s_c^t) + \frac{\partial J(W,b)}{\partial s_c^{t+1}} a_\beta^{t+1} + \frac{\partial J(W,b)}{\partial z_\alpha^{t+1}} w_{\alpha c} + \frac{\partial J(W,b)}{\partial z_\alpha^{t+1}} w_{\alpha c} + \frac{\partial J(W,b)}{\partial z_\alpha^{t+1}} w_{\alpha c}$$

LSTM 单元

最常见的表述：



$$
\begin{aligned}
i_t &= \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right)\\
f_t &= \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right)\\
c_t &= f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right)\\
o_t &= \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right)\\
h_t &= o_t \tanh(c_t)
\end{aligned}
$$

Thanks.

GRU-Gated Recurrent Unit

Learning phrase representations using RNN encoder-decoder for statistical machine translation

K Cho, B Van Merriënboer, C Gulcehre… - arXiv preprint arXiv: …, 2014 - arxiv.org

Abstract: In this paper, we propose a novel neural network model called RNN Encoder-Decoder that consists of two recurrent neural networks (RNN). One RNN encodes a sequence of symbols into a fixed-length vector representation, and the other decodes the representation into another sequence of symbols. The encoder and decoder of the proposed model are jointly trained to maximize the conditional probability of a target sequence ...

被引用次数：969　相关文章　所有 19 个版本　引用　保存

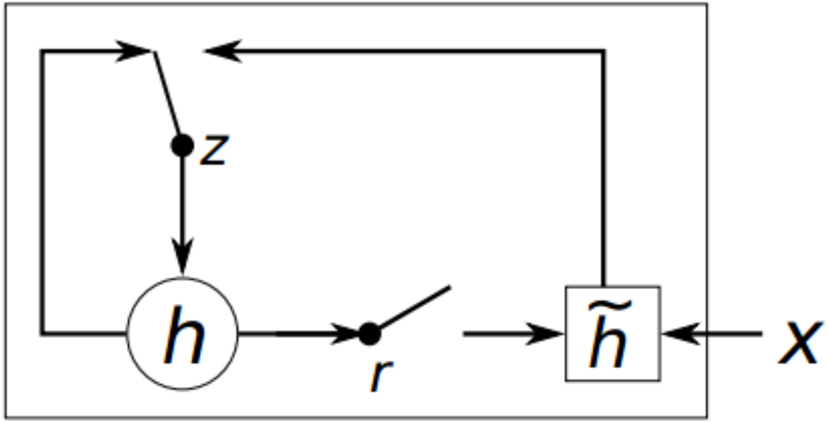


Figure 2: An illustration of the proposed hidden activation function. The update gate z selects whether the hidden state is to be updated with a new hidden state $\tilde{h}$. The reset gate r decides whether the previous hidden state is ignored. See Eqs. (5)–(8) for the detailed equations of r, z, h and $\tilde{h}$.

Depth Gated RNNs

Depth-gated LSTM

K Yao, T Cohn, K Vylomova, K Duh, C Dyer - arXiv preprint arXiv: …, 2015 - arxiv.org

... The idea of using **gated** linear dependence can also be used to connect the first layer memory cell c (1) t with the feature observation x (0) t . In this case **...** We varied the **depth** of **RNNs**. Results in Table 1 show that DGLSTM outperforms LSTM and GRU in all of the tested **depths**. **...**

Cited by 21 Related articles All 4 versions Cite Save

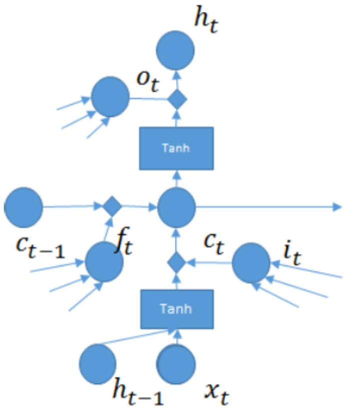
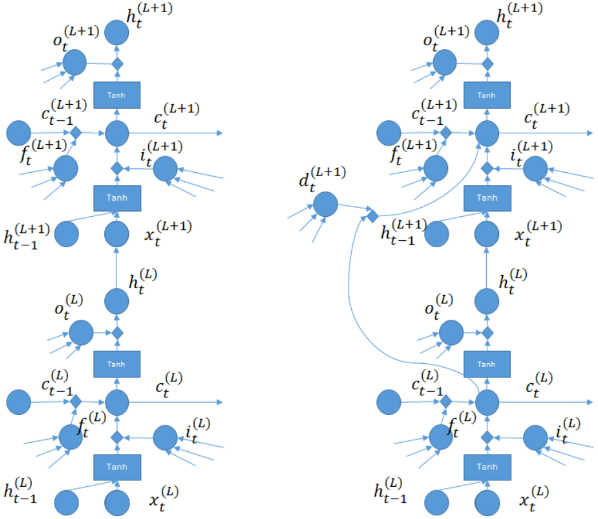


Figure 1: LSTM



(a) Stacked LSTM (b) Depth-gated LSTM

各种RNN模型的比较

LSTM: A search space odyssey

K Greff, RK Srivastava, J Koutník… - IEEE transactions on …, 2016 - ieeexplore.ieee.org

Abstract: Several variants of the long short-term memory (LSTM) architecture for recurrent neural networks have been proposed since its inception in 1995. In recent years, these networks have become the state-of-the-art models for a variety of machine learning problems. This has led to a renewed interest in understanding the role and utility of various computational components of typical LSTM variants. In this paper, we present the first ...

被引用次数：184　相关文章　所有 5 个版本　引用　保存

[PDF] An empirical exploration of recurrent network architectures

W Zaremba - 2015 - jmlr.org

Abstract The Recurrent Neural Network (RNN) is an extremely powerful sequence model that is often difficult to train. The Long Short-Term Memory (LSTM) is a specific RNN architecture whose design makes it much easier to train. While wildly successful in practice, the LSTM's architecture appears to be ad-hoc so it is not clear if it is optimal, and the significance of its individual components is unclear. In this work, we aim to determine ...

被引用次数：187　相关文章　所有 8 个版本　引用　保存　更多

Thanks.