RNN language model: speed up by factorization

Student: Shi Libin
Tutor: Rong wenge

Context

Introduction

Neural Language Model

Task: learn the pattern in human language, compute the probability of a sequence of words, $P(w_1, w_2, \dots, w_T)$

Statistic language model,

$$P(w_1, w_2, \dots, w_T)$$

$$= \prod_{i=1}^{T} P(w_i | w_1, \dots, w_{i-1})$$

Markov chain

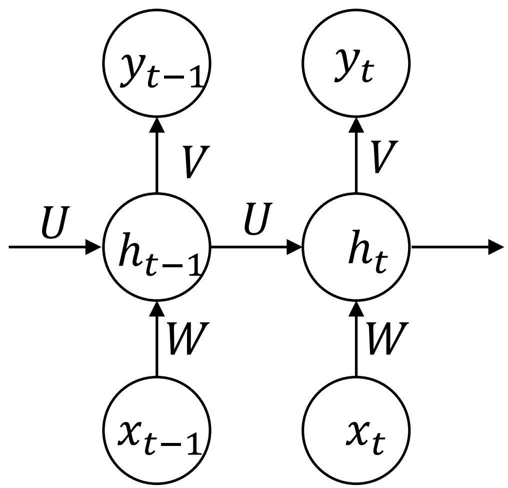
$$\approx \prod_{i=1}^{T} P(w_i | w_{i-(n-1),\dots,w_{i-1}})$$

Problem: the curse of dimension because the number of possible combinations of n words (n-gram) from a dictionary (e.g. 50,000 words) is immensely.

Introduction

Neural Language Model

Recurrent Neural network



$$h_t = \sigma(Uh_{t-1} + Wx_t + b)$$
$$y_t = softmax(Vh_t + c)$$

Good Performance!

- Fit the situation of LM

$$y_t = x_{t+1}$$
$$p(x_{t+1}|x_{\leq t}) = g_\theta(h_t)$$
$$h_t = \phi_\theta(x_t, h_{t-1})$$

- Reduce the size of model

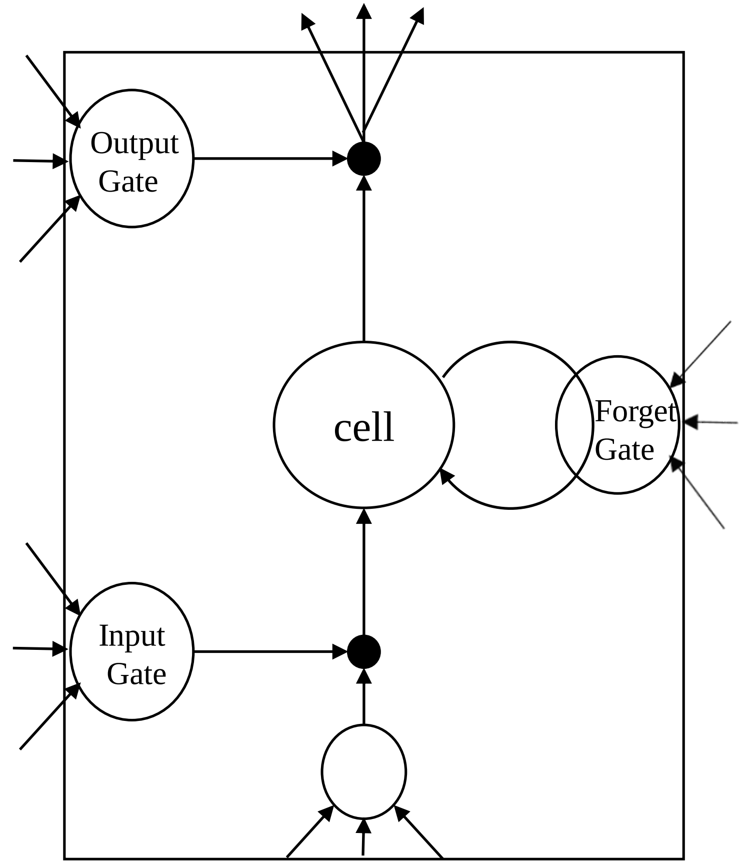
$$O(V) \ll O(V!)$$

Introduction

Neural Language Model

Long Short-Term Memory (LSTM)

$$i_t = \sigma\left(U^i h_{t-1} + W^i x_t + b^i\right)$$
$$f_t = \sigma\left(U^f h_{t-1} + W^f x_t + b^f\right)$$
$$g_t = tanh(U^g h_{t-1} + W^g x_t + b^g)$$
$$c_t = c_{t-1} \cdot f_t + g_t \cdot i_t$$
$$o_t = \sigma(U^o h_{t-1} + W^o x_t + b^o)$$
$$h_t = o_t \cdot \tanh(c_t)$$



Introduction

Metrics

Perplexity:

$$
\begin{aligned}
&PPL(w_1 \dots w_T) \\
&= 2^{-\frac{1}{T}\log_2 p(w_1 \dots w_T)} \\
&= \sqrt[T]{\frac{1}{p(w_1 \dots w_T)}} \\
&= \exp\left(-\frac{1}{T}\ln\big(p(w_1 \dots w_T)\big)\right) \\
&= \exp(loss)
\end{aligned}
$$

Others: Word Error Rate, Model size, Runtime

Related work

Problem of normal RNNLM: time consuming to calculate softmax

$$y_t = softmax(Wh_t + c)$$

$$y_{t,j} = P\left(x_{t+1} = v_j \middle| x_t, \dots, x_1\right) = \frac{\exp\left(W_j h_t + c\right)}{\sum_{k=1}^{|V|} \exp(W_k h_t + c)}$$

$$\frac{\partial}{\partial\theta} \log P_\theta^{x_{\le t}}(x_{t+1}) = \frac{\partial}{\partial\theta} s_\theta(x_{t+1}, x_{\le t}) - \sum_{x'} P_\theta^{x_{\le t}}(x') \frac{\partial}{\partial\theta} s_\theta(x', x_{\le t})$$

Sampling based models: Importance sampling (Bengio et al. , 2003), NCE (Mnih & Teh, 2012), Blackout (Ji et al., 2015)

Class level based models: two-level classed based softmax (Goodman, 2001), RNN with factorized output layer (Mikolov et al., 2011), hierarchical softmax (Morin & Bengio, 2005; Mnih & Hinton, 2009)

Related work

Sampling based models

Important sampling (IS)

$$\frac{\partial}{\partial\theta}\log P_\theta^{x_{\le t}}(x_{t+1})$$
$$= \frac{\partial}{\partial\theta}s_\theta(x_{t+1}, x_{\le t}) - \sum_{x'} P_\theta^{x_{\le t}}(x')\frac{\partial}{\partial\theta}s_\theta(x', x_{\le t})$$
$$\approx \frac{\partial}{\partial\theta}s_\theta(x_{t+1}, x_{\le t}) - \frac{1}{R}\sum_{j=1}^{m} r(x_j)\frac{\partial}{\partial\theta}s_\theta(x_j, x_{\le t})$$

Where $r(x) = \frac{\exp(s_\theta(x,contex))}{Q(w=x)}$ and $R = \sum_j^m r(x_j)$
Q is unigram distribution of training set.

Problem: the fewer samples we use, the worse is this approximation;
network's distribution P might diverge from Q during training.

Related work

Sampling based models

Noise Contrastive Estimation(NCE)

$$l_\theta = -\log P\left(D_{y_t} = 1 | x_t, x_{<t}\right) + \sum_{j=1}^{m} \log\left(P\left(D_{y_t} = 0 | \widetilde{x_t}^{j}, x_{<t}\right)\right)$$

$$P_\theta\left(D_{y_t} = 1 | x_t, x_{<t}\right) = \frac{\exp(h^T v'_x)}{\exp(h^T v'_x) + kQ(x)}$$

$$P_\theta\left(D_{y_t} = 0 | x_t, x_{<t}\right) = \frac{kQ(x)}{\exp(h^T v'_x) + kQ(x)}$$

Advantage: stable and fast for training
Disadvantage: still time consuming at test time

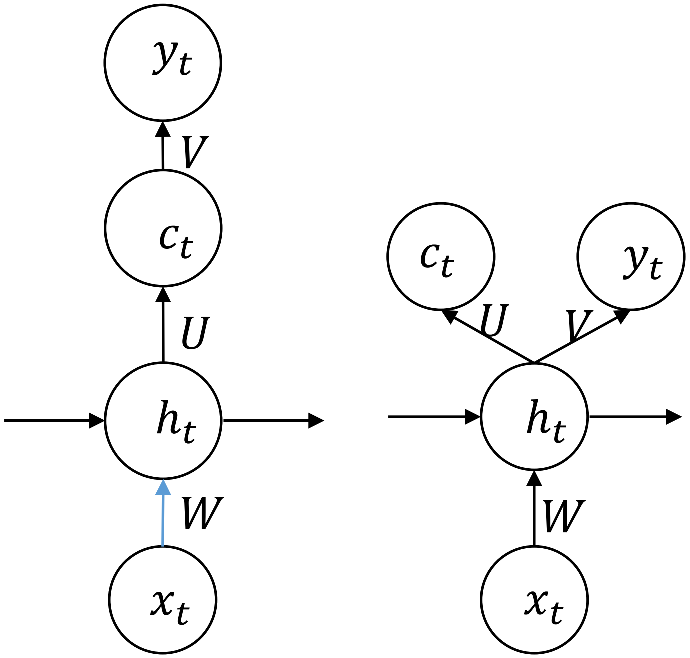
Related work

Factored level based models

RNN with output factorized by class layer

According to ***(Goodman, 2001)***
$P(y_t|context)$
$= P(c_t|context)P(y_t|c_t)$
$O(|H| * |V|) \rightarrow O(|H| * |C|)$

In (Mikolov et al., 2011),
$P(y_t|context)$
$= P(c_t|h_t)P(y_t|c_t, h_t)$

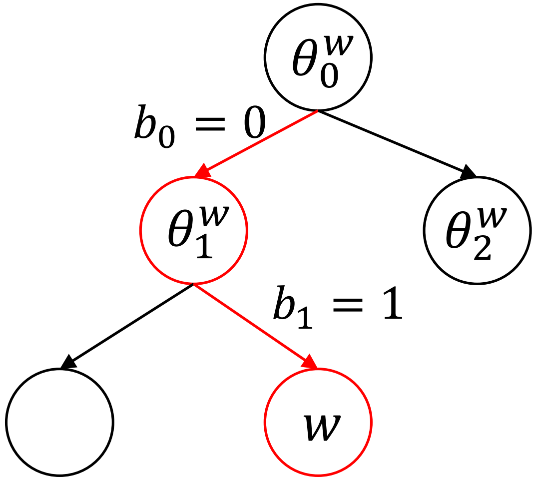


Advantage: reduce computational complexity,
Disadvantage: huge size of embedding layer

Related work

Factored level based models

Tree Hierarchical Softmax



$$P(w|x_{\le t}) = \prod_{j=1}^{m} P\left(b_j(w)|b_0(w), \dots, b_{j-1}(w), x_{\le t}\right)$$

$$P\left(b_j = 1|Node_j, x_{\le t}\right) = \sigma\left(\theta_j^w h_t\right)$$
$$P\left(b_j = 0|Node_j, x_{\le t}\right) = 1 - \sigma\left(\theta_j^w h_t\right)$$

Advantage: $O(|H| * |V|) \rightarrow O(|H| * \log_2 |V|)$
Disadvantage: huge size of embedding layer

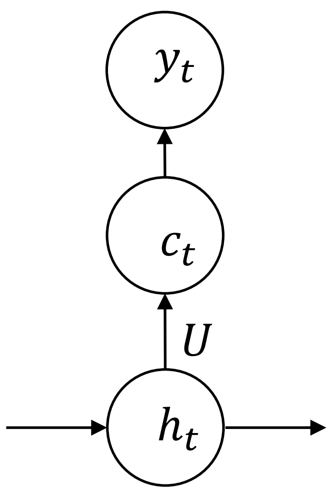
(Bengio, 2005), (Hinton, 2009)

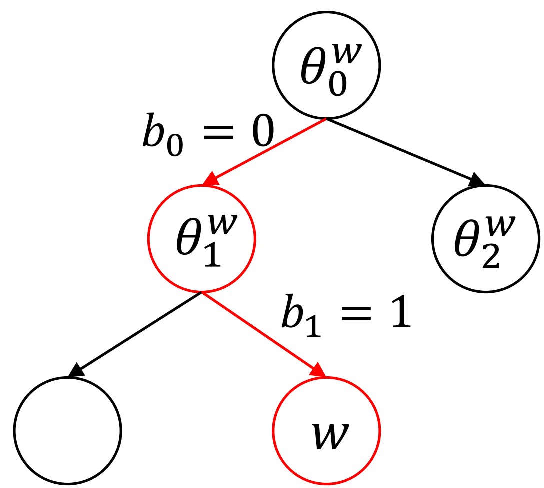
Related work

Factored level based models

Problem: how much deeper of **Hierarchical Softmax**

“It is inefficient to matrix-multiplication when one of the dimensions is small. This observation suggests that hierarchical organizations of words with a low number of children per node, such as binary Huffman codes, are highly suboptimal”
[Efficient Softmax Approximation for GPUs] (Grave, 2016)





lightRNN

1. Shared embedding word table

1-D word list

Index	word
x_1	the
x_2	i
…	…
x_{100}	weekend
x_{101}	week

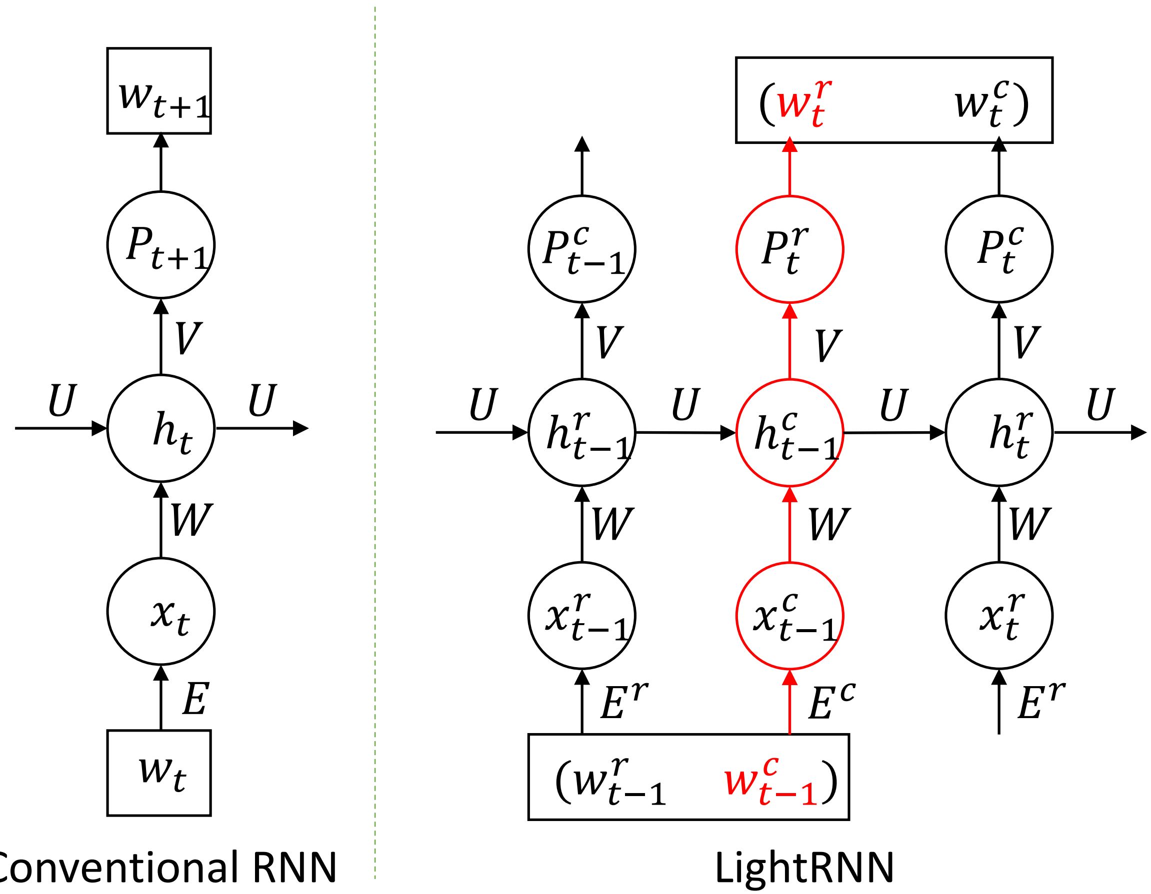
2-D word table

Index	word
(x_1^r, x_1^c)	the
(x_1^r, x_2^c)	i
…	…
(x_2^r, x_1^c)	weekend
(x_2^r, x_2^c)	week

Index	x_1^c	x_2^c	x_3^c
x_1^r	the	i	…
x_2^r	weekend	week	…
x_3^r	…	…	…

lightRNN

2. lightRNN model



$$
\begin{aligned}
&P(w_t) \\
&= P(w_t^r) * P(w_t^c) \\
&= P_t^r * P_t^c \\
&= s(h_{t-1}^c V) * s(h_t^r V)
\end{aligned}
$$

lightRNN

2. lightRNN model

Theano code

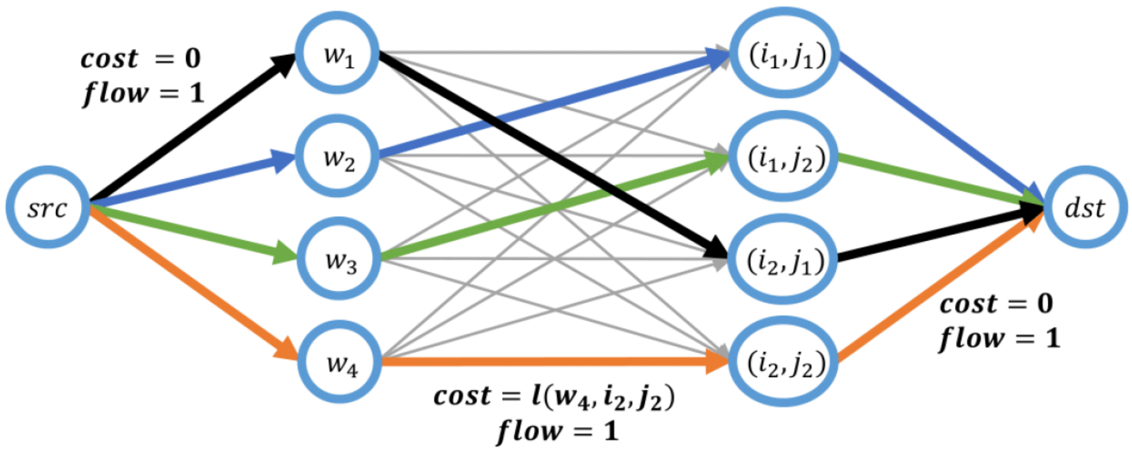
```
def recurrence(x_t, m_r, m_c, h_tml_r, h_tml_c):
    x_er = self.Exr[x_t[:, 0], :]
    x_ec = self.Exc[x_t[:, 1], :]

    concated_r = T.concatenate([x_er, h_tml_c], axis=-1)
    h_t_r = self.f(T.dot(concated_r, self.W) + self.b)
    h_t_r = h_t_r * m_r[:, None]
    concated_c = T.concatenate([x_ec, h_t_r], axis=-1)
    h_t_c = self.f(T.dot(concated_c, self.W) + self.b)
    h_t_c = h_t_c * m_c[:, None]
    return h_t_r, h_t_c

[h_r, h_c], update = theano.scan(fn=recurrence,
                sequences=[self.x, self.mask_r, self.mask_c],   # x.shape() = (n_maxlen, n_batch, 2)
                outputs_info=[dict(initial=T.zeros((self.n_batch, self.n_hidden))),
                        dict(initial=T.zeros((self.n_batch, self.n_hidden)))],
                truncate_gradient=-1)
```

lightRNN

3. MCMF for word re-allocation

$$NLL\big(w, r(w), c(w)\big)$$
$$= \sum_{w \in S_w} -\log P(w) = \sum_{w \in S_w} -\log P_r(w) + \sum_{w \in S_w} -\log P_c(w)$$

Minimum cost maximum flow (MCMF) algorithm



Using google/or-tools (cost-scaling push-relabel algorithm)

Analysis of lightRNN

LightRNN can be classified as Factored level based models

$$
\begin{aligned}
&P(w_t|w_{<t}) \\
&= P(w_t^r|w_{<t}^r, w_{<t}^c) \; * P(w_t^c|w_{\le t}^r, w_{<t}^c) \\
&= P(class|context) * P(index|class)
\end{aligned}
$$

row 832	Karwan Narok Cocodrie Noja Anambra Alaska. Lantau Willmar Zululand Tianmen ...
row 852	281-211 3-6-0 17-of-44 21-for-27 100-64 1,173-767 10-to-2 7-and-5 15,350 of-15 ...
row 861	103-run 12-way 23-hit 151-game 13-ball 105-meter 302-minute 189-yard 67-foot ...
row 872	totaled hunted rigged scored vetoed inflicted froze swam won dried raged smiled ...
row 877	plods riles hankers misbehaves contrives utilizes disbands computes propagates ...
row 887	www.angiotech.com www.huntsman.com media.floridarealtors.org 2010.census.gov ...
row 889	years. decade evening hours. weeks spring summer. day-and-a-half April-to-June ...
row 891	44kg 63pc 170mph 18cm 22C 12A 150bp 17st 656ft 2Mbps 680g 10x 13ph. 2M ...

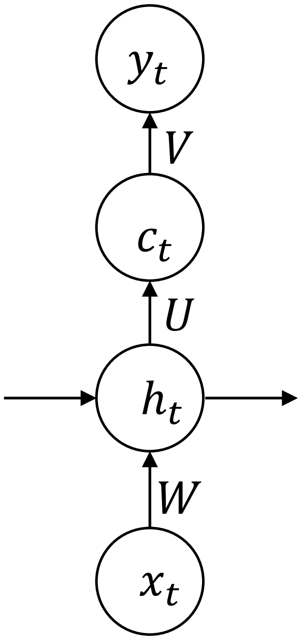
Analysis of lightRNN

Speed up by factored layer

$$
\begin{aligned}
&P(w_t|w_{<t}) \\
&= P(w_t^r|w_{<t}^r, w_{<t}^c) \ * P(w_t^c|w_{\le t}^r, w_{<t}^c) \\
&= P(class|context) * P(index|class)
\end{aligned}
$$

If size of vocabulary is 10,000 and is placed into a 100*100 word table, $O(|V|) \rightarrow O\left(2\sqrt{|V|}\right)$

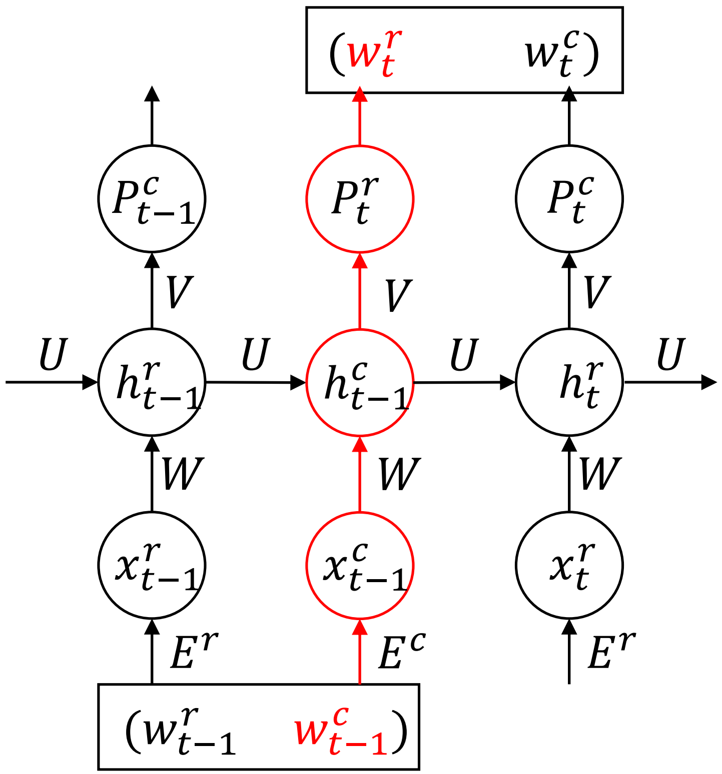
Factorization both for input layer and output layer!



cHSM

Analysis of lightRNN

Conclusion of lightRNN



Advantage:

- Speed up training process by a factor of 2
- Reduce the model size by a factor of around 50

Problem:

- Increase the recurrent steps of RNN by a factor of 2
- 2-D word table may be a limit

Some ideas

Utilize class information for re-allocation

Make use of unbalance of row and column and row index is seen as class information.

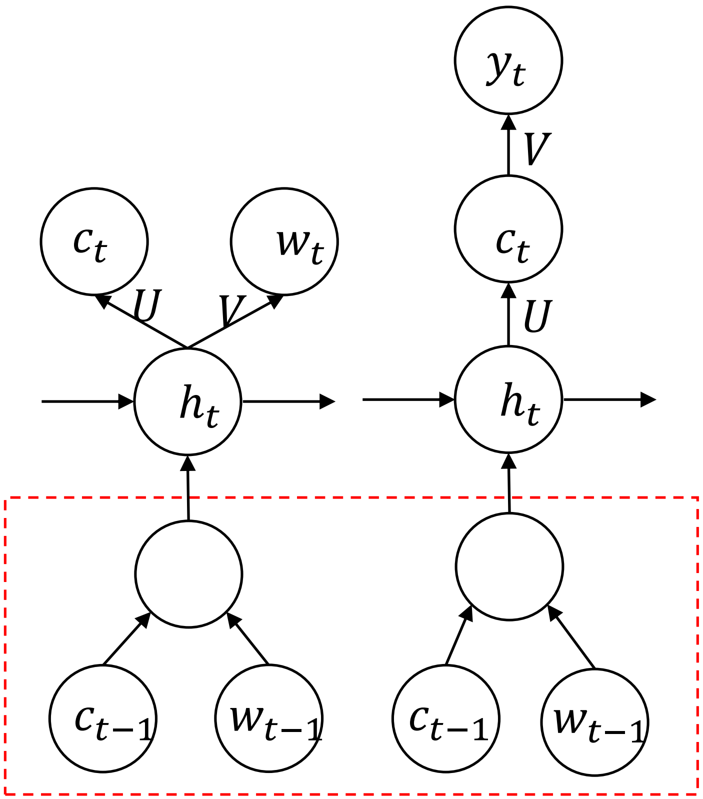
Only re-allocate on row index

Computational complexity of MCMF
$O(K|V|^2) \rightarrow O(K|V|^{3/2})$

Some ideas

Factorized RNN (f-RNN)

Encoding factorial class information into input layer

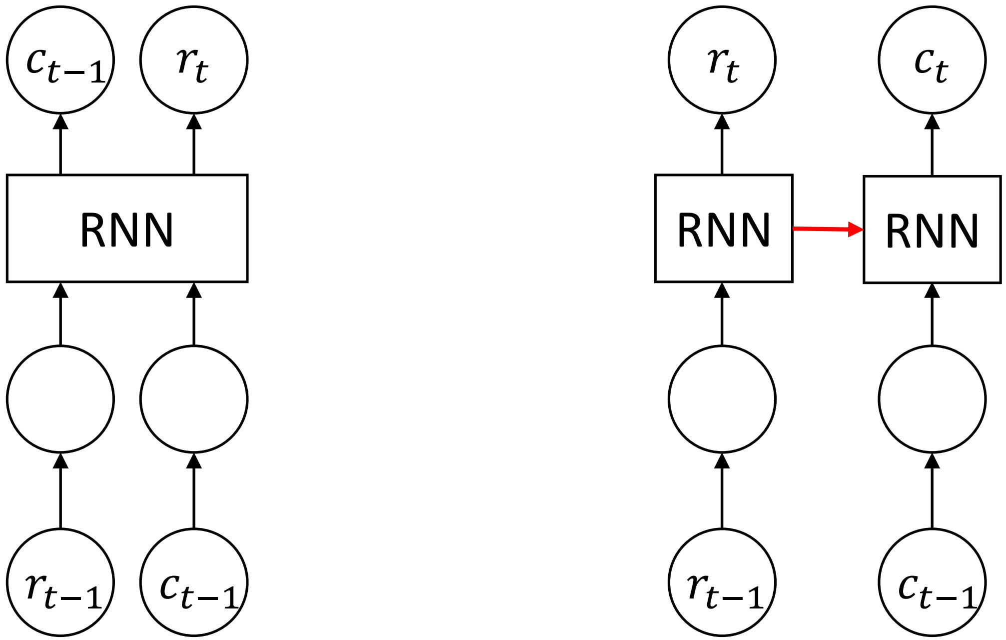


- Class information is determined before inputting RNN
- Need a structure to recognize and focus factored information of input layer

Some ideas

Factorized RNN (f-RNN)

Encoding factorial class information into RNN layer

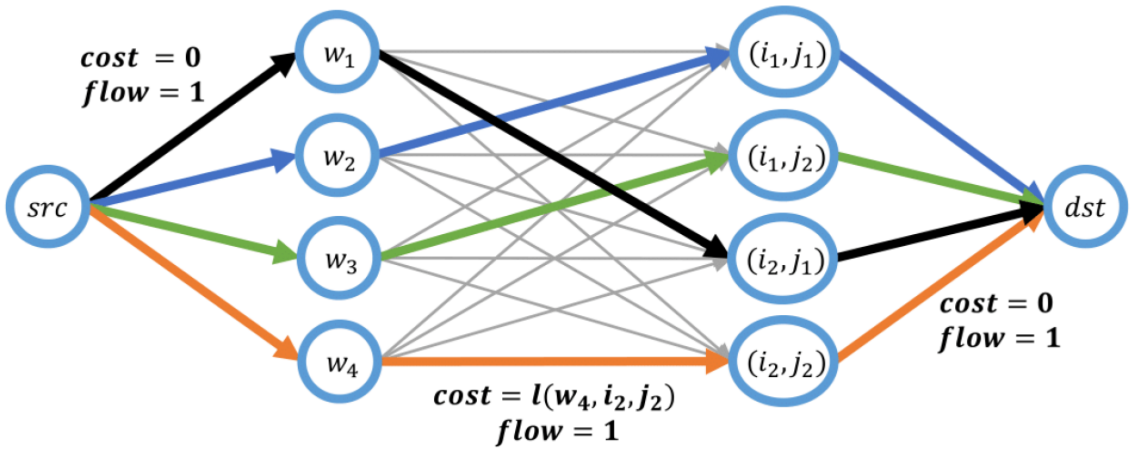


lightRNN

Experiments

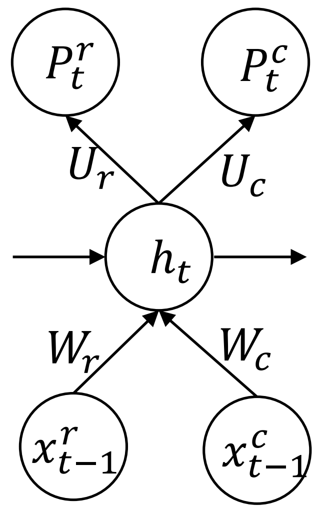
Dataset	PTB			WikiText-103			One Billion		
	train	valid	test	train	valid	test	train	valid	Test
Sentences	42,068	3,370	3,761	1,801,350	3,760	4,358	29,994,193	306,835	613,376
Vocabulary	10,000			267,735			788,996		
Best result (PPL)	114			130 (8 epoch)			123 (1 epoch)		

Experiments



MCMF on row index			
Algorithm	PPL	Re-allocation full time	Solve time
MCMF	114	836s	205s
row-MCMF	117	11s	2s

Experiments



Algorithm	PPL	Runtime/epoch
rnnlm	114	620s
f-rnnlm	245	220s

Conclusion

To be continue

- Improve the implementation of lightRNN
- Study in detail factored RNN
- Implement f-RNN
- Evaluate each Initialize method

Thanks for your attention