IJCAI-17 MELBOURNE

BEIHANG UNIVERSITY

Exploration of Hierarchical Softmax for Neural Language Models

Nan Jiang, Wenge Rong, Min Gao, Yikang Shen and Zhang Xiong

Beihang University, Chongqing University and Universite de Montreal

Softmax with Over-large Vocabularies

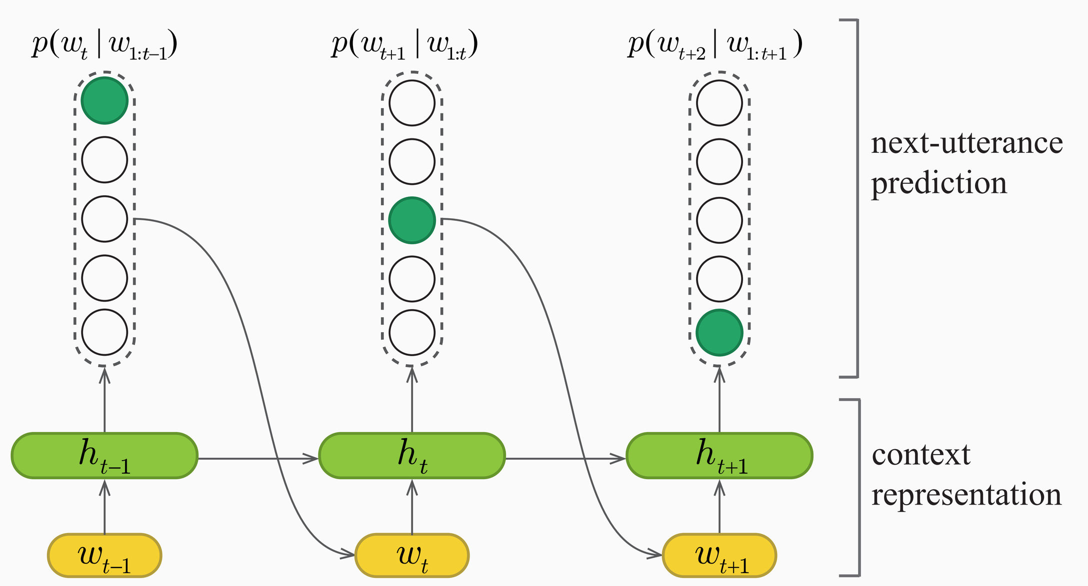


Figure: Recurrent Neural Language Model.

$$\log p(w_1, \cdots, w_T) = \sum_{t=1}^{T} \log p(w_t|w_{1:t-1}), \quad p(w_i|h) = \frac{\exp(h^\top v_{w_i})}{\sum_{w_j \in \mathcal{V}} \exp(h^\top v_{w_j})} \quad (1)$$

- **Vocabulary Truncation:** the vocabulary are split into a short-list of the most frequent words and a tail of OOV words.
- **Sampling based Approximation:** compute only a tiny fraction of the outputs dimensions to achieve computational efficiency, like Noise contrastive estimation, Importance sampling, Blackout sampling and etc.
- **Vocabulary Factorization:** decompose the flattened architecture of vocabulary into a class structure (i.e., class-based hierarchical softmax) or hierarchical binary tree (i.e., tree-based hierarchical softmax).

Parallelised Tree-based Hierarchical Softmax

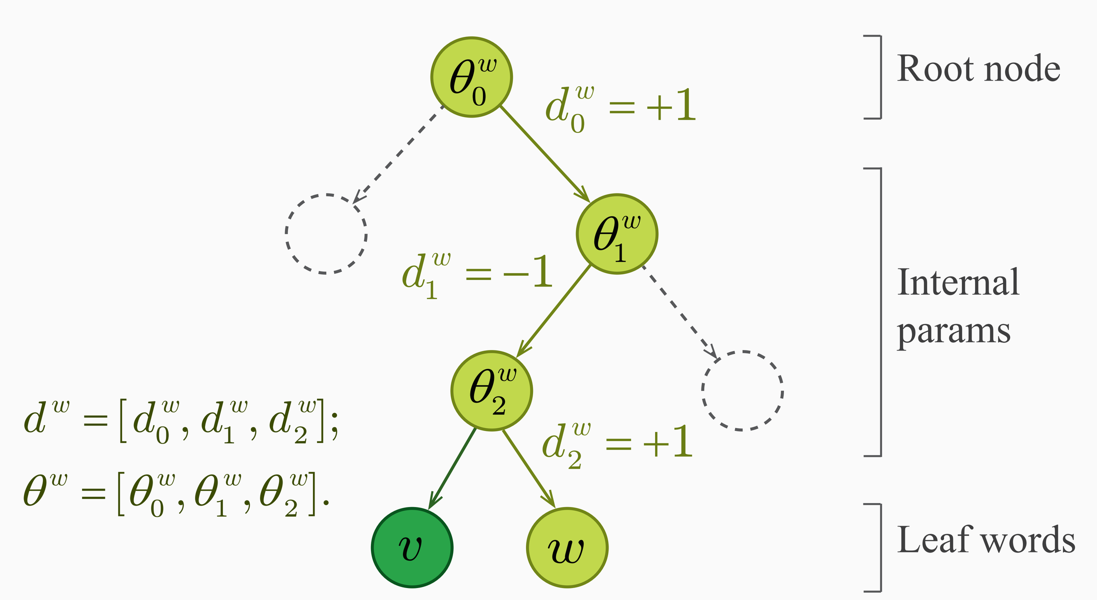


Figure: Tree-based Hierarchical Softmax. Internal nodes are parameterised by θ_i^w, edges between nodes are parameterised by d_i^w, where d^w is a vector and θ^w is a matrix.

probability for internal node θ_i^w:

$$p(d_i^w = \pm 1|\theta_i^w, h) = \sigma(d_i^w \theta_i^w h) \quad (2)$$

log probability for word w:

$$\log p(w|h) = \log \prod_{i=0}^{l^w-1} p(d_i^w|\theta_i^w, h) = \sum_{i=0}^{l^w-1} \log \sigma(d_i^w \theta_i^w h) = \log \sigma(d^{w\top} \theta^w h) \quad (3)$$

A major advantage: it avoids normalise the probability over the whole vocabulary, as the summarised probabilities of words in the tree equals to one.

$$\sum_{w \in \mathcal{V}} p(w|h) = \sum_{w \in \mathcal{V}} \sum_{i=0}^{l^w-1} \sigma(d_i^w \theta_i^w h) = 1. \quad (4)$$

Why it is faster?

Parallelised tree-based cost function:

$$\ell(\theta|h, w) = -\log \prod_{i=0}^{l^w-1} \sigma(d_i^w \theta_i^w h) = -\log \sigma(d^{w\top} \theta^w h) = \zeta(-d^{w\top} \theta^w h) \quad (5)$$

Conventional cost function:

$$\ell'(\theta|h, w) = \sum_{i=0}^{l^w-1} \{(1 - d_i'^w) \log(\sigma(\theta_i^w h)) + d_i'^w \log(1 - \sigma(\theta_i^w h))\} \quad (6)$$

Two Main difference:

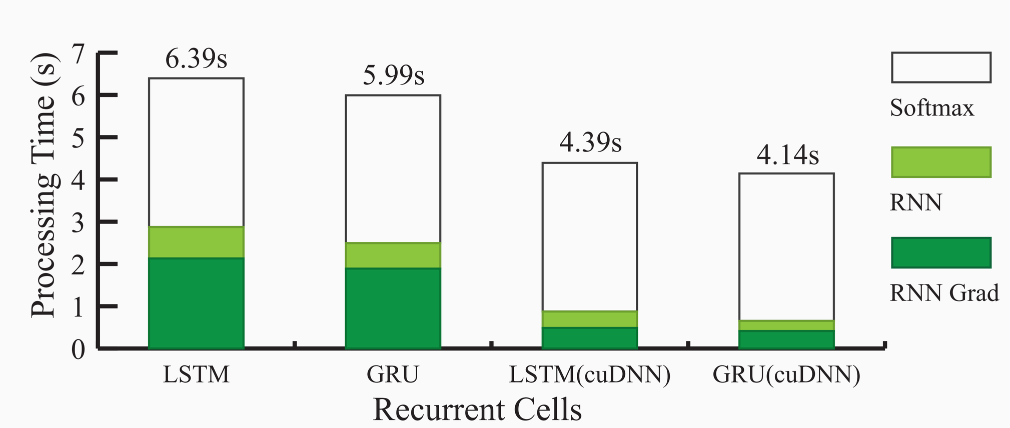
- tHSM algorithm involves many tiny matrix multiplications, instead in p-tHSM we load all parameters (d^w, θ^w) directly as 1D vector and 2D matrix at the expense of runtime memory consumption and we consider the multiplications of this vector and giant matrix;
- A compact loss function of the model is deducted and the nodes' log-probability are calculated simultaneously which results in better time efficiency for p-tHSM model.

Experiment setups

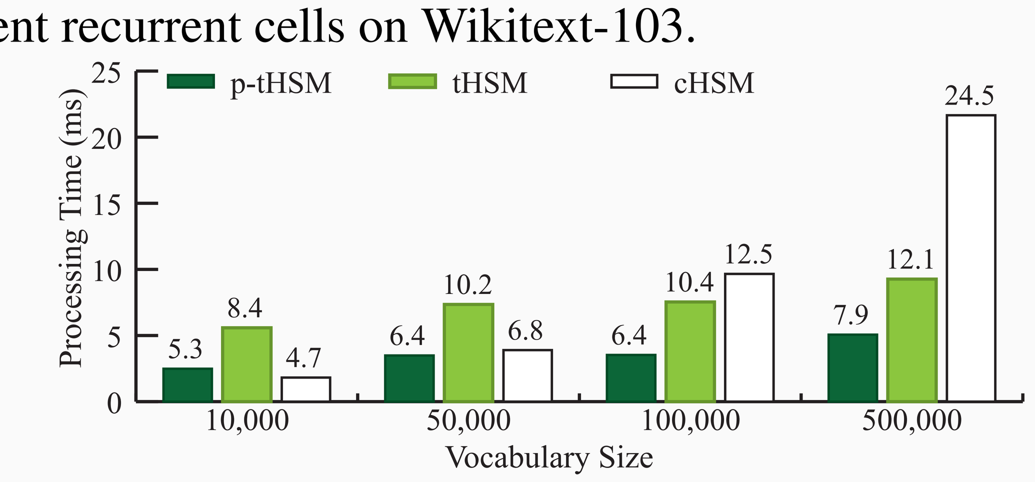
Dataset Statistics: Number of tokens for train, valid and test, vocabulary size, and fraction of out-of-vocabulary rate.

	PTB			WikiText-2			WikiText-103		
	train	valid	test	train	valid	test	train	valid	test
Articles	2,000	155	155	600	60	60	28,475	60	60
Sents	42,068	3,370	3,761	36,718	3,760	4,358	1,801,350	3,760	4,358
Vocab	10,000			33,278			267,735		
OOV	4.8%			2.6%			0.4%		

Runtime benchmark



Large Vocab Problem: Calculation Time of three modules with different recurrent cells on Wikitext-103.



Scalability: cHSM, tHSM and p-tHSM algorithms with different vocabulary size.

Speed Comparison: Time and memory comparison on GPUs and CPUs with WikiText-103 dataset.

	Runtime memory	Total (ms) cpu	Total (ms) gpu	Forward (ms) cpu	Forward (ms) gpu
Softmax	$\lvert\mathcal{H}\mathcal{V}\rvert$	510.4	262.1	352.2	62.9
cHSM	$2\lvert\mathcal{H}\rvert\sqrt{\lvert\mathcal{V}\rvert}$	506.5	**40.6**	28.7	14.6
tHSM	$\lvert\mathcal{H}\rvert$	1,004.0	444.4	8.1	5.6
p-tHSM	$\lvert\mathcal{H}\rvert \log \lvert\mathcal{V}\rvert$	**383.5**	86.4	**7.0**	**1.4**

Perplexity Results

Word Clustering Strategy Analysis: apply different clustering methods to initialise the distribution of words over the tree.

Methods	Valid	Test
Random Shuffle	199.62	189.37
Alphabetical Order	154.02	149.12
Huffman Clustering	134.33	129.34
Brown Clustering	**133.12**	**128.78**

All Experiments Benchmark: Perplexity benchmark on validation and testing dataset with PTB, WikiText-2 and WikiText-103 corpus.

	PTB		WikiText-2		WikiText-103	
	Valid	Test	Valid	Test	Valid	Test
GRU + Softmax	**131.59**	**125.10**	**169.07**	**160.45**	170.19	171.02
GRU + NCE	139,79	137.35	210.19	189.15	194.78	195.01
GRU + Blackout	137.68	135.49	201.51	185.31	192.11	193.76
GRU + cHSM	133.17	125.05	179.64	169.09	171.81	166.74
GRU + p-tHSM + Huffman	134.33	129.34	218.42	216.05	165.70	166.11
GRU + p-tHSM + Brown	133.12	128.78	186.23	189.58	**164.15**	**161.55**

Our Contributions

- We propose a parallelised mathematical model for modeling vanilla hierarchical softmax algorithm, making it compatible with GPUs;
- Aiming at improving its stability, we employ several n-gram based and semantic clustering algorithms to initialise the word hierarchy before the training stage;
- We conducted empirical analysis and comparisons on the standard PTB, WikiText-2 and WikiText-103 datasets with other traditional optimisation methods to assess its efficiency and accuracy on GPUs and CPUs.