Exploration of Tree-based Hierarchical Softmax for Recurrent Language Models

Nan Jiang[†‡], Wenge Rong[†‡], Min Gao[§], Yikang Shen[♯], Zhang Xiong[†‡]
[†] State Key Laboratory of Software Development Environment, Beihang University, China
[‡] School of Computer Science and Engineering, Beihang University, China
[§] School of Software Engineering, Chongqing University, China
[♯] Montréal Institute for Learning Algorithms, Universté de Montréal, Canada
{nanjiang, w.rong, xiongz}@buaa.edu.cn, gaomin@cqu.edu.cn, yi-kang.shen@umontreal.ca

Abstract

Recently, variants of neural networks for computational linguistics have been proposed and successfully applied to neural language modeling and neural machine translation. These neural models can leverage knowledge from massive corpora but they are extremely slow as they predict candidate words from a large vocabulary during training and inference. As an alternative to gradient approximation and softmax with class decomposition, we explore the tree-based hierarchical softmax method and reform its architecture, making it compatible with modern GPUs and introducing a compact tree-based loss function. When combined with several word hierarchical clustering algorithms, improved performance is achieved in language modelling task with intrinsic evaluation criterions on PTB, WikiText-2 and WikiText-103 datasets.

1 Introduction

Language modeling is a basic and fundamental task in Computational Linguistics (CL), and have attracted much academic and commercial attention for the last few decades. The use of Language Model (LM) is universal in applications such as Neural Machine Translation (NMT) [Jean *et al.*, 2015] and Automatic Speech Recognition [Wang and Wang, 2016].

Specifically, language model is composed of two parts: *context* representation and *next-step word* prediction. The *context* knowledge is generally represented by several existing neural-based architectures, e.g. Feedforward Neural Network [Bengio *et al.*, 2003], Recurrent Neural Network (RNN) [Mikolov *et al.*, 2010], and Character-level Convolution Neural Network (CharCNN) [Kim *et al.*, 2016]. Besides, the *next-step word* prediction, usually represented by softmax algorithm, involves computing the probability of a true word in the whole vocabulary.

Nonetheless, intensive computations are involved in language model with modern neural network during training and inference. Particularly, computing the softmax probability normalisation function exploits great portion of time, and this issue has become a bottleneck for all variations and applications as the practical speed decreases dramatically with the vocabulary grows [Chen *et al.*, 2016].

Therefore, various studies have attempted to address this challenge in literature. An intuitive method was vocabulary truncation, which relies on maintaining a shortlist of the most frequently word-level or subword-level units, thereby increases the speed of normalisation and expectation based on the truncated vocabulary [Schwenk, 2007; Sennrich *et al.*, 2016]. Subsequently, several other workarounds were proposed, which can be roughly divided into sampling-based approximations and factored output layer methods.

Sampling-based methods avoid intensive computation by computing only a tiny fraction of the vocabulary. E.g., Noise Contrastive Estimation (NCE) regards multi-class identification tasks as psudo logistic classification by classifying the empirical word distribution from noise words' distribution [Gutmann and Hyvärinen, 2012]. Subsequently, Blackout sampling is proposed to overcome NCE algorithm's context dependence problems [Ji *et al.*, 2016]. Although these approximations can speed up training significantly, the normalisation term have to evaluated during inference.

Alternatively, factored output layer decompose the original flattened architecture into a class structure or hierarchical tree, i.e., class-based (cHSM) and tree-based Hierarchical Softmax (tHSM) respectively [Chen *et al.*, 2016]. cHSM method partition the vocabulary into mutually exclusive classes, and tHSM builds nested categories of words to form a binary tree with words on the leaves. Recently, the cHSM method has been benchmarked with different clustering strategies and compared with other sampling-based approaches [Chen *et al.*, 2016]. Besides, the historical proposed tHSM method that builds upon WordNet knowledge or Huffman coding performs relatively better than the cHSM method in term of time efficiency [Mikolov *et al.*, 2013b].

After investigating the tHSM model, it was shown that it calculates the internal probability step-by-step and layer-by-layer, giving a time complexity of $\mathcal{O}(\log |\mathcal{V}|)$ [Mikolov *et al.*, 2013b; Mnih and Hinton, 2009]. Considering the modern architecture of GPUs and generalised matrix-matrix operations, it is interesting to ask whether it is possible to calculate the internal probability in parallel? Furthermore, the Huffman coding scheme for tHSM only consider the word's frequency statistics while words' context and semantic information have yet been exploited, which demands for detailed discussion of word hierarchical clustering's strategies to initialise the words' distribution over the tree.

Therefore, in this research, we propose a parallelised mathematical model for modeling tHSM's variants (i.e., p-tHSM). Moreover, hierarchical Brown clustering and other possible clustering algorithms are employed to initialise the word hierarchy before the training stage, to improve the stability and performance of p-tHSM algorithm. Furthermore, we conducted empirical analysis and comparisons on the standard Penn Tree Bank (PTB) [Marcus *et al.*, 1993], WikiText-2 and WikiText-103 text datasets [Merity *et al.*, 2017] with other conventional optimisation methods to assess its efficiency and accuracy on GPUs and CPUs[1].

2 Background

Softmax with Over-large Vocabularies. As a standard probability normalisation method for multi-label classification, the softmax function and its gradient can be formalised as:

$$\begin{aligned} p(w_i|h) =& \frac{\exp(h^\top v_{w_i})}{\sum_{w_j \in \mathcal{V}} \exp(h^\top v_{w_j})} \\ \frac{\partial p(w_i|h)}{\partial v_{w_j}} =& p(w_j|h)(\delta_{ij} - p(w_i|h))h^\top \end{aligned} \tag{1}$$

where h denotes the hidden layer (i.e., *context* representation), v_w is the target word-embedding of word w and δ_{ij} represents the *Kronecker delta*.

The forward probability propagation and backward gradient optimisation manipulating all the words in the target vocabulary, thereby resulting in low efficiency. For a vocabulary comprising $|\mathcal{V}|$ words, the overall time complexity is $\mathcal{O}(|\mathcal{V}|)$. This computational burden is relatively high even for modern architectures of GPUs, which are highly suitable for matrix multiplication with its parallelism.

In order to alleviate the computational bottleneck when computing the normalisation term in Eq. 1, various approximations have been developed, which can be divided into three categories: vocabulary truncation, sampling-based approximation, and factored output layer methods.

Vocabulary Truncation. Intuitively, a relatively smaller vocabulary list can be maintained to avoid computing a large vocabulary, where it will run faster with less memory consumption. Maintaining a short-list of the most frequent words and pruning the rest has been explored in literature [Marcus *et al.*, 1993], while excessive words in texts will be mapped to "$\langle$unk$\rangle$" tokens. Another point of view is to split words into subword-level units according to its character n-gram statistics to reduce its vocabulary size [Sennrich *et al.*, 2016].

Sampling-based Approximation. Sampling-based approaches have been employed successfully to approximate the softmax and its gradient over a large vocabulary in numerous fields [Baltescu and Blunsom, 2015; Mnih and Kavukcuoglu, 2013]. The core idea is to compute only a tiny fraction of the outputs dimensions to achieve computational efficiency.

To be specific, NCE approximation regards the multi-class prediction problem as psudo binary classification and this method employs an auxiliary loss to optimise the goal of maximizing the probability of correct words, while also minimising the noise probability [Mnih and Teh, 2012]. To be specific, the model learn to classify w_0 from $\{w_1 \cdots w_k\}$, where w_0 is the empirical example and $\{w_1 \cdots w_k\}$ are noise samples generated from a prior unigram distribution $q(w)$. The normalised probability of positive categories and the joint probability of k noise samples is:

$$\begin{aligned} \tilde{p}(y=1|h) =& \frac{\exp(h^\top v_{w_0})}{\exp(h^\top v_{w_0}) + k * q(w_0)} \\ \tilde{p}(y=0|h) =& \prod_{i=1}^{k} \frac{k * q(w_i)}{\exp(h^\top v_{w_i}) + k * q(w_i)} \end{aligned} \tag{2}$$

where $\tilde{p}(y=0|h)$ involves a sum over k noise samples instead of a sum over the entire vocabulary, making the NCE training time linear in the number of noise samples $\mathcal{O}(k)$ and independent of the vocabulary size.

In addition, the recently proposed Blackout sampling algorithm implicitly combine the advantages of importance sampling and NCE to solve the unigram distribution's context dependency problems in NCE algorithm [Ji *et al.*, 2016].

Overall, these approximations significantly accelerate the training speed but time is still required to sample abundant noises with the unigram distribution $q(w)$. Nonetheless, the partition function have to evaluated during inference as candidates are predicted in the whole vocabulary [Gutmann and Hyvärinen, 2012].

Factored Output Layer. Factored output layer can dramatically reduce the cost of representing the probability distribution as well as learning and inference, and several proposed structures can be divided into two categories: cHSM and tHSM methods.

For cHSM model, it transforms one-step multi-label prediction into two-step multi-label prediction, where the former predicts the class of the word and the latter predicts the probability of this word in the corresponding class, as shown in Fig.1 and Eq.3. Therefore, this procedure employs cascaded softmax prediction to avoid the multiplication of a large vocabulary. If we partition the vocabulary into k classes $\{c_1, \cdots, c_k\}$, such that $c_i \bigcap c_j = \phi, i \neq j$ and $\mathcal{V} = \bigcup_{i=1}^{k} c_i$, then the cHSM model can be written as:

$$p(w_{ij}|h) = p(c_i|h) \cdot p(w_{ij}|c_i, h) \tag{3}$$

where w_{ij} belongs to class c_i. If each class contains $\sqrt{|\mathcal{V}|}$ words, the optimal time complexity can be reduced to $\mathcal{O}(\sqrt{|\mathcal{V}|})$ [Goodman, 2001].

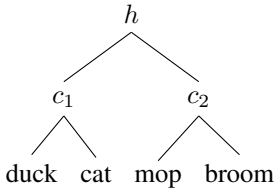

Figure 1: Class-based Hierarchical Softmax.

[1]All our codes and models are publicly available at `https://github.com/jiangnanhugo/lmkit`

Likewise, tHSM decomposes one-step multi-class classification into several steps of logistic classification so the vocabulary is organised as binary tree, where words lie on its the leaves and the intermediate nodes are internal parameters. In the case of a balanced tree structure, the optimal case can be $\mathcal{O}(\log|\mathcal{V}|)$. Traditionally, tree can be constructed from WordNet with human experts [Morin and Bengio, 2005] or Huffman coding with word unigram distribution [Mikolov *et al.*, 2013b]. Yet, expert knowledge is expensive in real world challenge and the Huffman coding scheme only considers the unigram statistics while words' context, syntactic or semantic information are neglected.

3 Methodology

In this section, we begin by defining the *word polarity encoding* scheme for modeling tree-based parameters and structures. With this scheme, we derive a compact and tight form of cost function as well as its gradient for the proposed parallelised tHSM (p-tHSM) model. Before training the p-tHSM model, we investigate several word hierarchical clustering strategies to define the distribution of word over the tree, in order to achieve stable performance.

3.1 Word Polarity Encoding

In the case of binary tree with words on its leaves, we can locate every word by all visiting the internal nodes from root to the leaf. Here, the route for word w denotes the internal nodes θ_i^w and edges d_i^w it visits.

To illustrate, θ_i^w represents the non-leaf nodes on the i^{th} layer on the route to word w, and $\theta_i^w \in \mathbb{R}^m, i \in [0, l^w - 1]$. Likewise, d_i^w represents the edge that connects the $(i-1)^{th}$ and i^{th} layer's node. For each non-leaf node, moving down to the left branch is labelled as -1 and selecting the right branch is labelled as $+1$. So, $d_i^w \in \{-1, +1\}, i \in [0, l^w - 1]$. Besides, $l^w \approx \log|\mathcal{V}|$ denotes the route's length from the root to the leaf word. With this schema, we can change the traditional word index or one-hot representation into a word polarity encoding tuple (d^w, θ^w) for word w by denoting the polarity route to locate each word.

During implementation, we maintain a path looking-up table Γ, memorising all the visited internal nodes' index for each word w from root to leaf. Such that, θ^w is retrieved from parameter matrix Θ by selecting all the nodes from $\Gamma(w)$. Since the first dimension of Θ is $\mathcal{V}-1$ (as $\sum_{i=0}^{\log \mathcal{V}} 2^i = |\mathcal{V}|-1$), no external parameters are involved for training. Besides, d^w is retrieved by obtaining the w^{th} row vector from matrix $\mathcal{D}$, where w is the index in the vocabulary. Furthermore, $\{\Gamma, \mathcal{D}\}$ are defined by the word clustering hierarchy, which is introduced in the following section, and Θ is optimised by gradient descent on the training dataset.

3.2 Tree-based Loss and Gradient

During every step in the target word tree, we make a logistic prediction about whether go to the left branch or right part at each non-leaf node. The probability of i-th label $d_i^w \in \{-1, 1\}$ given the i-th node and hidden layer h is:

$$p(d_i^w = \pm 1 | \theta_i^w, h) = \sigma(d_i^w \theta_i^w h) \tag{4}$$

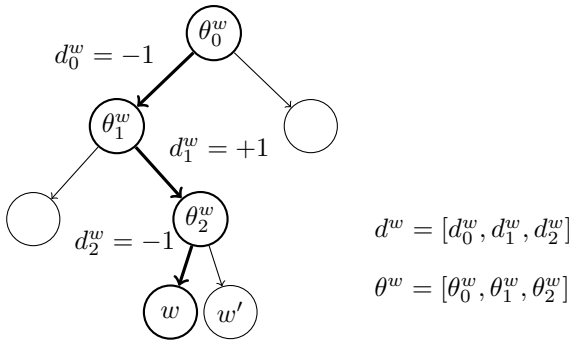

Figure 2: Tree-based Hierarchical Softmax. Internal nodes are parameterised by θ_i^w, edges between nodes are parameterised by d_i^w, where d^w is a vector and θ^w is a matrix. E.g., $d^w = [-1, +1, -1]$ and $d^{w'} = [-1, +1, +1]$.

where $\sigma(z) = 1/(1+\exp(-z))$ denotes the sigmoid function and its symmetric rule is employed for abbreviation [Kleinbaum and Klein, 2010]: $\sigma(z) + \sigma(-z) = 1$. Therefore, the probability of word w is the joint product of the probability of the route (d^w, θ^w) taken from the root to the corresponding leaf node:

$$\begin{aligned} \log p(w|h) &= \log \prod_{i=0}^{l^w-1} p(d_i^w | \theta_i^w, h) \\ &= \sum_{i=0}^{l^w-1} \log \sigma(d_i^w \theta_i^w h) \\ &= \log \sigma({d^w}^\top \theta^w h) \end{aligned} \tag{5}$$

and the corresponding loss function of the model $\mathcal{L}(\theta|h, w)$ is defined by the negative log-likelihood:

$$\begin{aligned} \mathcal{L}(\theta|h, w) &= -\log \prod_{i=0}^{l^w-1} \sigma(d_i^w \theta_i^w h) \\ &= -\log \sigma({d^w}^\top \theta^w h) \\ &= \log(1 + \exp(-{d^w}^\top \theta^w h)) \\ &= \zeta(-{d^w}^\top \theta^w h) \end{aligned} \tag{6}$$

where $\zeta(z)$ denotes the softplus function: $\zeta(z) = \log(1 + \exp(z))$ and its gradient is sigmoid-like function: $\mathrm{d}\zeta(z)/\mathrm{d}z = \sigma(z) = 1/(1 + \exp(-z))$ [Dugas *et al.*, 2000]. Minimising the negative log-likelihood is directly maximising softplus loss and the probability of estimated words.

In the traditional tHSM algorithm, the model calculates the log-probability of every node layer-by-layer consequently the overall joint log-probability of this route is summarised linearly, thus the time complexity of tHSM takes $O(\log \mathcal{V})$:

$$\begin{aligned} \mathcal{L}'(\theta|h, w) = \sum_{i=0}^{l^w-1} \{&(1 - d_i'^w) \log(\sigma(\theta_i^w h)) \\ &+ d_i'^w \log(1 - \sigma(\theta_i^w h))\} \end{aligned} \tag{7}$$

where $d_i'^w \in \{0, 1\}$, and these two parts model the joint probability of the left and right branch of internal nodes separatively. Noticeably, the main difference between p-tHSM and

tHSM is that: a) tHSM algorithm involves many tiny matrix multiplications, instead we load all parameters (d^w, θ^w) directly as 1D vector and 2D matrix at the expense of memory consumption and we consider the multiplications of this vector and giant matrix, as shown in Fig 2; b) Based on that, 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.

Likewise, the model's parameters $\{\theta^w, h\}$ are optimised with regard to its gradient.

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \theta^w} &= (\sigma(d^{w\top}\theta^w h) - 1)d^{w\top}h \\ \frac{\partial \mathcal{L}}{\partial h} &= (\sigma(d^{w\top}\theta^w h) - 1)d^{w\top}\theta\end{aligned} \tag{8}$$

A major advantage of the binary decomposition is that it avoids normalise the probability over the whole vocabulary, as the summarised probabilities of words in the tree is naturally equals to one.

$$\sum_{w\in\mathcal{V}} p(w|h) = \sum_{w\in\mathcal{V}} \sum_{i} \sigma(d_i^w \theta_i^w h) = 1 \tag{9}$$

In the procedure of inference, for a node in the i-th layer, we choose the left branch when $p(d_i^w|\theta_i^w, h) \geq 0.5$ and select the right child when the opposite applies. Thus, the computational time complexity is $\mathcal{O}(\log|\mathcal{V}|)$.

3.3 Word Hierarchy Criterion

The polarity encoding of each word (d^w, θ^w) in the vocabulary is closely related to the tree's structure. In the proposed method, we employ several clustering algorithms to improve the performance of p-tHSM algorithm.

Random Shuffle. This intuitive method ignores the word hierarchy. Thus, words are randomly located in the leaves of the tree's nodes. This is the worst case for revealing the lower bound of the tree modelling method.

Alphabetical Order. It is more efficient to sort the vocabulary of words in alphabetical order and words with similar tokens for strings share similar meanings in the text. Thus, similar clustering results might be obtained to the commonly used frequency binning method.

Huffman Clustering. This method sorts the vocabulary based on word frequency and can be directly applied on p-tHSM model, where words distribution on the tree are generated by the Huffman clustering rules. So frequent words share similar short path and rare words are assigned with similar long path [Mikolov *et al.*, 2013b].

Brown Clustering. The Brown algorithm is an agglomerative hierarchical clustering method that clusters words to maximise the mutual information for bigram words [Brown *et al.*, 1992; Liang, 2005], The binary prefix string for every word it generates can be applied to initialise p-tHSM model.

4 Experimental Study

In order to validate the proposed p-tHSM method and compare with other baseline methods, we conducted experimental study and analysis with recurrent language modeling task on three standard text datasets with several metrics.

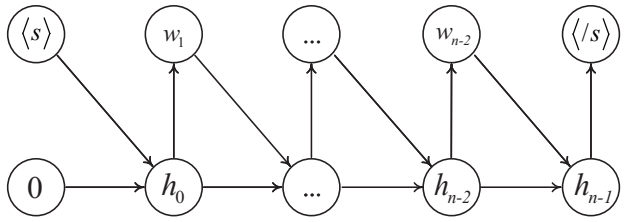

Figure 3: Recurrent Neural Language Model. Every sentence is wrapped with start (i.e., $\langle s \rangle$) and end (i.e., $\langle$ /s$\rangle$) tokens. Before predicting the next-step word w_{t+1}, the input is received from the last hidden state h_{t-1} and current word w_t.

4.1 Recurrent Language Model and Metrics

The goal of a language model is to learn the probability for a sequence of words. In particular, given a sequence of T words: $[w_1, w_2 \cdots, w_T]$, the probability of this sequence can be decomposed into the joint product of the conditional probability using the chain rule:

$$p(w_1, w_2 \cdots, w_T) = \prod_{t=1}^{T} p(w_t|w_{<t}) \tag{10}$$

where $p(w_t|w_{<t})$ denotes the next-step word probability given its previous context $[w_1, \cdots, w_{t-1}]$ as inputs, and its usually modeled by RNN model, as shown in Fig. 3. In this research, recurrent neural network with gated units (GRU) [Chung *et al.*, 2014] is preferred as it requires a much smaller set of parameters to model the internal recurrence mechanism as well as its gating function ease the gradient vanishing problem [Hochreiter, 1998].

In the training process, we regard the cross-entropy between the true word distribution and the predicted distribution, also known as the negative log-likelihood, as the model's loss function:

$$\mathcal{L}(\theta|w_1, \cdots, w_T) = -\frac{1}{T}\sum_{t=1}^{T} \log p(w_t|w_{<t}) \tag{11}$$

A standard score used for comparing LM's performance is the model's *perplexity* ($\mathcal{PP}$), which denotes the degree of confusion when choosing the next-step word among candidates and a lower perplexity for a language model implies better predictability. The *perplexity* is the exponential of the average negative log-likelihood: $\mathcal{PP}(w_1, \cdots, w_T) = \exp(\mathcal{L}(\theta))$, denoting that we are directly optimising the *perplexity* metric during training.

In addition, the *training time efficiency, vocabulary scalability* and *runtime memory consumption* should also be considered as important metrics to benchmark models. Thus, we analysed these evaluative metrics for different optimisations algorithms in both theoretical and empirical perspectives on GPUs and CPUs respectively. Finally, experiments are conducted and results are collected on the standard PTB, WikiText-2 and WikiText-103 dataset to evaluate their accuracy and efficiency, and the detailed statistics of these datasets[2] are demonstrated in Table 1.

[2]https://metamind.io/research/the-wikitext-long-term-dependency-language-modeling-dataset/

Table 1: Statistics of the PTB, WikiText-2 and WikiText-103 Dataset.

Dataset	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
Sentences	42,068	3,370	3,761	36,718	3,760	4,358	1,801,350	3,760	4,358
#Vocabulary		10,000			33,278			267,735	
Out-of-Vocabulary (%)		4.8%			2.6%			0.4%	

4.2 Results and Discussions

Factored Output Layer Comparison. As shown in Table 2, we benchmarked the empirical time complexity and parameters size for training the same batched data on GPUs and CPUs. The "Softmax" method denotes the traditional flattened softmax method, the "cHSM" algorithm is the class-based hierarchical softmax, the "tHSM" method denotes the traditional tree-based hierarchical softmax that has been discussed in section 3, and the "p-tHSM" algorithm is the proposed parallelised tree-based hierarchical softmax.

We tried to process the WikiText-103 dataset with these algorithms and calculated the average time required for processing one batch data. Besides, the input sentence's max length, hidden layer, output vocabulary and batch size were set as $\{50, 256, 267735, 20\}$, respectively. Furthermore, the "Total" process denotes the process for the forward propagation and backward gradient optimisation, and the "Forward" process denotes the time consumption required from the input of the data until the model's cost was calculated.

Subsequently, we calculated the runtime memory usage for the above mentioned algorithms, as shown in Table 2. Here, the "Runtime" denotes the required memory we need to load during training, and $|\mathcal{H}|$ is the embedding dimension. During training, tHSM only consumed a minimal memory resource while p-tHSM covered a much larger set of memory, and p-tHSM employed larger memory and achieved better speedup ratio when considering both memory consumption and speed.

In order to verify the scalability of cHSM, tHSM and p-tHSM algorithms with relevance to the vocabulary size, results are collected in Fig. 4. In order to visualise the impact of the p-tHSM algorithm, the "Softmax" method is not included as it consumed much more computational time compared with the others. Clearly, the cHSM scales with the square root of the vocabulary size (i.e., $\mathcal{O}(\sqrt{|\mathcal{V}|})$) whereas p-tHSM exhibits stable performance as the vocabulary size increases.

Based on these experiments, we may conclude that the proposed p-tHSM method outperformed the historical benchmark $\mathcal{O}(\log|\mathcal{V}|)$ and achieved a satisfactory speed-up ratio for the hierarchical softmax architecture. This performance was attributable to the acceleration based on the hardware's parallelism, but also due to the fundamental structure of the p-tHSM method, which could maintain the target word tree in a parallel manner.

Word Clustering Strategy Analysis. Chen et al. revealed that the performance of cHSM is sensitive to the hierarchical clustering of the words based on the tree [Chen *et al.*, 2016]. Likewise, we considered several existing tree clustering criteria in order to stabilise the performance of p-tHSM algorithm.

Table 2: Runtime time and memory comparison on GPUs and CPUs with WikiText-103 dataset.

	Runtime	Total (ms)		Forward (ms)	
	memory	cpu	gpu	cpu	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**

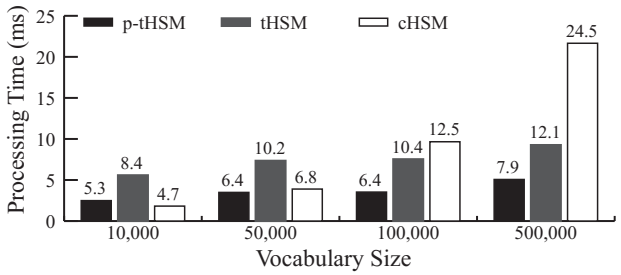

Figure 4: Scalability of cHSM, tHSM and p-tHSM algorithms with relevance to the vocabulary size.

Table 3: Perplexity of p-tHSM method with various hierarchical clustering algorithms on PTB Dataset.

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**

So we compared several tree clustering criteria using the PTB dataset in Table 3. Randomly shuffling words and splitting them into clusters of equal sizes performed the worst as it did not provide any information about the distribution of words. Besides, Huffman clustering (i.e., frequency binning) method categorised words into frequency groups, where words in the same branches of the tree shared a similar words' frequency [Mikolov *et al.*, 2013a]. This method performed better than alphabetical order clustering algorithm, which words were sorted considering their alphabetical order and split them into equal-size groups of words. Although Brown clustering token time to calculate and similar words were grouped together according to its bigrams' distribution, it performed the best than the others. To be noticed, Randomly shuffling and alphabetical order methods partition the vocabulary into balanced tree structures while the rest not.

Table 4: 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 [Gutmann and Hyvärinen, 2010]	139,79	137.35	210.19	189.15	194.78	195.01
GRU + Blackout [Ji *et al.*, 2016]	137.68	135.49	201.51	185.31	192.11	193.76
GRU + cHSM [Chen *et al.*, 2016]	133.17	125.05	179.64	169.09	171.81	166.74
GRU + p-tHSM (pretrain via Huffman)[Mikolov *et al.*, 2013b]	134.33	129.34	218.42	216.05	165.70	166.11
GRU + p-tHSM (pretrain via Brown) [Brown *et al.*, 1992]	133.12	128.78	186.23	189.58	**164.15**	**161.55**

All Experiments Benchmark. As shown in Table 4, we collected all the perplexity results on validation and testing dataset of the above mentioned three standard corpus for every algorithms. Besides, all experiments implemented with Theano framework [Theano Development Team, 2016] were run on one standalone GPU device with 12 GB of graphical memory (i.e., Nvidia K40m), which allowed the large embedding matrix multiplications to be possible. However, the resource of one GPU memory was exploited quickly with the rise of parameters' dimension.

To be noticed, we employed one layer of GRU cell as the *context* representation for all these algorithms, the dimension of which was set to 256. Besides, for the NCE and Blackout approximations, the hyper-parameter k was set to $|\mathcal{V}|/20$ for smaller PTB and Wikitext-2 datasets and $k = |\mathcal{V}|/200$ for the larger WikiText-103 dataset. Furthermore, for cHSM method, we partitioned the vocabulary according its words' unigram distribution.

Considering results on PTB and Wikitext-2 datasets, the original softmax achieved the best ever score than the rest algorithms, as it did not introduce any structural loss in cHSM and p-tHSM algorithms or sampling-based variational loss in Blackout and NCE approximations.

For the last ever-large Wikitext-103 dataset, the p-tHSM method with Brown clustering not only achieved better results than the Huffman clustering methods, but also performed better than the others. Alternatively, the cHSM model were capable of getting similar results than the p-tHSM variants, denoting we might achieved better results with other suitable clustering algorithms for cHSM methods. Even the Wikitext-103 and Wikitext-2 dataset shared the same testing set, we found the original softmax were hard to converge and performed the worst results. Besides, for sampling-based methods, it converged to much better results than the softmax method at the same time improve the time efficiency.

To conclude, after replacing the traditional Huffman coding scheme in tHSM with a word polarity coding scheme and a compact tree-based model p-tHSM is achieved, we demonstrated that this novel proposed coding scheme allowed the calculations to run in parallel on GPUs. This reduced the time complexity of raw tHSM $\mathcal{O}(\log|\mathcal{V}|)$ and obtained the optimum speed-up ratio for large vocabulary problems. Furthermore, after testing several existing clustering algorithms for stabilising the performance of the p-tHSM model, we found that word clustering based on the tree model was closely related to binary classification based on the internal nodes.

5 Conclusions and Future Work

In this study, we considered the over-large vocabulary problem in the language modeling fields. In the literature, various approaches have been proposed to address this issue, which can be roughly divided into three categories: vocabulary truncation, sampling-based approximation, and factored output layer algorithms.

The first approach was easy to implement yet not perfectly solve the problem as it can not generalise to large real world tasks. While the second categories of method can reduce the training time consumption in an efficient manner without summarising all the words through variations of importance sampling, but it failed during inference because a prior distribution is applied and the multi-polynomial sampling method was not time efficient. The third methods changed the flatten architecture of the softmax algorithm into a heuristic hierarchical structure with two possible types: cHSM and tHSM algorithms. The cHSM method built on the two-step softmax classification method and tHSM extended this idea to $\log|\mathcal{V}|$ steps binary classification tasks.

In this research, we not only proposed a potential encoding scheme to improve the standard tHSM but also built a compact mathematical model for tHSM's variants (i.e., p-tHSM) but also exploited the advantages of GPU hardware, so it can efficiently calculate the exact softmax and gradient with a large target vocabulary. Remarkably, the complexity of this algorithm is beyond the historical record $\mathcal{O}(\log|\mathcal{V}|)$ but not in constant time complexity, thereby allowing it to tackle very large vocabulary problems. Furthermore, we evaluated several word hierarchical clustering algorithms for organizing words in the tree model in a more efficient manner. Compared with the baselines, the results showed that the speed-up ratio was enhanced for the other probability normalisation method and more efficient tree clustering was obtained.

In future research, we plan to explore the application of the softplus activation function and optimise several existing hierarchical clustering algorithms, in order to design a more efficient hierarchical structure.

Acknowledgments

This work was partially supported by by the State Key Laboratory of Software Development Environment of China (No. SKLSDE-2017ZX-16), the National Natural Science Foundation of China (No. 61332018), and the Fundamental Research Funds for the Central Universities.

References

[Baltescu and Blunsom, 2015] Paul Baltescu and Phil Blunsom. Pragmatic neural language modelling in machine translation. In *Proc. of NAACL-HLT*, pages 820–829, 2015.

[Bengio *et al.*, 2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[Brown *et al.*, 1992] Peter F. Brown, Vincent J. Della Pietra, Peter V. de Souza, Jennifer C. Lai, and Robert L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

[Chen *et al.*, 2016] Wenlin Chen, David Grangier, and Michael Auli. Strategies for training large vocabulary neural language models. In *Proc. of ACL*, pages 1975–1985, 2016.

[Chung *et al.*, 2014] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[Dugas *et al.*, 2000] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. Incorporating second-order functional knowledge for better option pricing. In *Proc. of NIPS*, pages 472–478, 2000.

[Goodman, 2001] Joshua Goodman. Classes for fast maximum entropy training. In *Proc. of ICASSP*, pages 561–564, 2001.

[Gutmann and Hyvärinen, 2010] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proc. of AISTATS*, pages 297–304, 2010.

[Gutmann and Hyvärinen, 2012] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13:307–361, 2012.

[Hochreiter, 1998] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(2):107–116, 1998.

[Jean *et al.*, 2015] Sébastien Jean, KyungHyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. In *Proc. of ACL*, pages 1–10, 2015.

[Ji *et al.*, 2016] Shihao Ji, S. V. N. Vishwanathan, Nadathur Satish, Michael J. Anderson, and Pradeep Dubey. Blackout: Speeding up recurrent neural network language models with very large vocabularies. In *Proc. of ICLR*, 2016.

[Kim *et al.*, 2016] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models. In *Proc. of AAAI*, pages 2741–2749, 2016.

[Kleinbaum and Klein, 2010] David G Kleinbaum and Mitchel Klein. Maximum likelihood techniques: An overview. *Logistic regression*, pages 103–127, 2010.

[Liang, 2005] Percy Liang. *Semi-supervised learning for natural language*. PhD thesis, MIT, 2005.

[Marcus *et al.*, 1993] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[Merity *et al.*, 2017] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *Proc. of ICLR*, 2017.

[Mikolov *et al.*, 2010] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Proc. of Interspeech*, pages 1045–1048, 2010.

[Mikolov *et al.*, 2013a] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[Mikolov *et al.*, 2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*, pages 3111–3119. 2013.

[Mnih and Hinton, 2009] Andriy Mnih and Geoffrey E Hinton. A scalable hierarchical distributed language model. In *Proc. of NIPS*, pages 1081–1088. 2009.

[Mnih and Kavukcuoglu, 2013] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Proc. of NIPS*, pages 2265–2273. 2013.

[Mnih and Teh, 2012] Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. In *Proc. of ICML*, 2012.

[Morin and Bengio, 2005] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Proc. of AISTATS*, 2005.

[Schwenk, 2007] Holger Schwenk. Continuous space language models. *Computer Speech & Language*, 21(3):492–518, 2007.

[Sennrich *et al.*, 2016] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proc. of ACL*, 2016.

[Theano Development Team, 2016] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *CoRR*, abs/1605.02688, 2016.

[Wang and Wang, 2016] ZhongQiu Wang and DeLiang Wang. A joint training framework for robust automatic speech recognition. *IEEE/ACM Transactions on Audio, Speech & Language Processing*, 24(4):796–806, 2016.