Exploration of Tree-based Hierarchical Softmax for Recurrent Language Models

Abstract

Recently, variants of neural networks for computational linguistics have been proposed and applied successfully, such as recurrent language model and neural machine translation. These models can learn 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, which only speeds up during training, we explore the traditional class-based and tree-based hierarchical softmax methods. When combined with several word hierarchical clustering algorithms, we achieve improved performance in language modelling tasks with intrinsic evaluation criterions on PTB, WikiText-2 and WikiText-103 datasets.

1 Introduction

In the field of computational linguistics, language modelling (LM) is a basic and fundamental task, which is universal in applications such as neural machine translation (NMT) [Jean *et al.*, 2015] and automatic speech recognition [Wang and Wang, 2016].

In particular, LM has two key components: *context* representation and *next-step word* prediction. The *context* of LM can be represented by several existing architectures, e.g. feed forward neural network [Bengio *et al.*, 2003], recurrent neural network (RNN) [Mikolov *et al.*, 2010], or character-level convolution neural network [Kim *et al.*, 2016]. Besides, *next-step word* prediction involves computing the probability of a true word in the whole vocabulary, and it is usually represented by softmax method.

However, intensive computations are involved in LM with modern neural network during training and inference (i.e., the testing procedure), particularly in the *next-step word* prediction process, which is also called the over-large vocabulary challenge [Chen *et al.*, 2014]. Moreover, the issue of a large vocabulary has become a bottleneck for all variations and applications of LM as the practical training speed decreases dramatically with the vocabulary grows [Chen *et al.*, 2016].

Various studies have attempted to address this challenge and an early method was vocabulary truncation, which relies on the maintenance of a short-list of the most frequently words, thereby increasing the speed of normalization and expectation based on the vocabulary [Schwenk, 2007]. Several other workarounds were proposed subsequently, which can be roughly divided into sampling-based approaches and factored output layer methods.

Sampling-based approaches compute only a tiny fraction of the output's dimensions. For example, the importance sampling (IS) algorithm tries to estimate the model's gradient with a prior distribution [Bengio and Senecal, 2008]. Similarly, noise contrastive estimation (NCE) changes multi-class identification tasks into psudo logistic classification tasks by classifying the empirical word distribution from the distribution of noise words [Gutmann and Hyvärinen, 2012]. These approximations can speed up training significantly but they fail during inference.

Alternatively, factored output layer methods 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]. Recently, the cHSM method is 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 WorldNet knowledge or huffman coding performs relatively better than the cHSM in term of time efficiency [Morin and Bengio, 2005; 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, thereby giving a total complexity of $\log \mathcal{V}$ [Grave *et al.*, 2016]. 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 of tHSM only consider the word frequency while word's context and semantic information are ignored, which demands for detailed discussing of word hierarchical clustering.

In this study, we propose a generalised mathematical model for tHSM's variants and a parallelised tHSM (p-tHSM), which can reduce the time complexity to constant $\mathcal{O}(\mathcal{C})$. Furthermore, we discuss brown clustering with existing strategies for constructing word hierarchical clusters in order to improve the stability and performance of p-tHSM [Derczynski *et al.*, 2015]. To evaluate the proposed p-tHSM method, we conducted empirical analyses and comparisons on the stan-

dard PTB, WikiText-2 and WikiText-103 language datasets with other conventional optimization methods in order to assess its efficiency and accuracy on GPUs.

The remainder of this paper is organised as follows. In Section 2, we review some important concepts and previous studies of large vocabulary optimisation with LM. In Section 3, we explain our contributions to the field of large-scale RNN training with p-tHSM. In Section 4, we present the results of the experimental study. In Section 5, we give our conclusions and suggest some possible areas for future research.

2 Background

2.1 Softmax with Over-large Vocabularies

As a standard probability normalization method for multi-class classification, the softmax function and its gradient can be defined as:

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

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

The forward probability propagation and backward gradient optimization 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 computation bottleneck when computing the normalization 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.

2.2 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 remainder has been explored in a previous study [Marcus *et al.*, 1993]. This idea is easy to implement but it has the limitation that an excessive number of words in texts are mapped to "⟨unk⟩" tokens with a larger corpus.

2.3 Sampling-based Approximation

Sampling-based approaches have been employed successfully to approximate the softmax and its gradient over a large vocabulary in various 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.

In particular, the IS algorithm selects a subset of negative targets to approximate the negative reinforcement part of softmax normalization with a biased unigram distribution, as shown by Eq. 2.

$$\frac{\partial \log p(w|h)}{\partial \theta} = \frac{\partial h^\top v_w}{\partial \theta} - \sum_{w' \in \mathcal{V}} p(w'|h) \frac{\partial h^\top v_w}{\partial \theta} \tag{2}$$

By contrast, NCE approximation regards the multi-class prediction problem as psudo binary classification and this method employs an auxiliary loss to optimize the goal of maximizing the probability of correct words, while also minimizing the noise probability [Gutmann and Hyvärinen, 2010; Mnih and Teh, 2012].

Thus, given the output of the hidden layer h, the model tries 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 can be rewritten as:

$$\begin{aligned} \tilde{p}(y=1|h) =& \frac{\exp(h^\top v_{w_0})}{\exp(h^\top v_{w_0}) + k * q(w_0)} \\ \tilde{p}(y=0|h) =& \prod_{i=1}^{k} \frac{k * q(w_i)}{\exp(h^\top v_{w_i}) + k * q(w_i)} \end{aligned} \tag{3}$$

The speed-up ratio for NCE approximation is $\mathcal{O}(\mathcal{V})/\mathcal{O}(k)$, which is closely related to the sample size k.

In addition, the recently proposed Blackout sampling algorithm implicitly combine the advantages of the two methods described above to solve the unigram distribution does not change with context-dependent information problems [Ji *et al.*, 2016].

In general, all these approximations significantly accelerate the training speed but time is still required to sample abundant noises in the word unigram distribution. Nonetheless, the partition function must be evaluated during inference.

2.4 Factored Output Layer

The main advantage of using a factored output layer is that it can dramatically reduce the cost of representing the probability distribution as well as learning and inference. Thus, several hierarchical softmax methods with different structures have been proposed and we divide these methods into two categories: cHSM and tHSM methods.

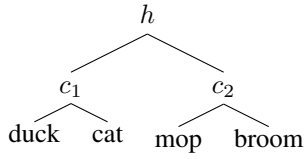

Figure 1: Class-based Hierarchical Softmax.

For one thing, cHSM 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. 4. Therefore, this procedure employs cascaded softmax prediction to avoid the multiplication of a

large matrix. 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 defined as:

$$p(w_{ij}|h) = p(c_i|h) \cdot p(w_{ij}|c_i, h) \tag{4}$$

where w_{ij} belongs to class c_i. If each class contains $\sqrt{\mathcal{V}}$ words, the time can be reduced from $\mathcal{O}(\mathcal{V})$ to $\mathcal{O}(\sqrt{\mathcal{V}})$.

For another, tHSM decomposes one-step multi-class classification into several steps of logistic classification so the vocabulary is organized as binary tree, where words lie on its the leaves and the intermediate nodes are internal variables. In the case of a balanced tree structure, the optimal case can be $\mathcal{O}(\log \mathcal{V})$. While, tHSM is modeled linearly and haven't exploit the specificities of modern GPUs architectures and matrix-matrix operations.

Moreover, tree can be constructed from WordNet with human experts [Morin and Bengio, 2005] or huffman coding with frequency binning [Mikolov *et al.*, 2013b]. Expert knowledge is expensive in real world challenge and the huffman coding scheme only consider the word frequency while word's context and semantic information are neglected.

3 Methodology

In this section, we describe the proposed p-tHSM method for addressing the time-consumption problem during training and inference, which can be divided into three steps: 1) defining a generalised *word polarity encoding* scheme for tHSM model; 2) with this scheme, we derive a compact and tight cost function that belongs to the spherical family; and 3) several word clustering strategies are applied to obtain better performance.

3.1 Polarity Encoding of Words

For a tree model with all the words on its leaves, we can locate each specific word based on the route it takes from the root to the leaf, where the route 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 $\theta_i^w \in \mathbb{R}^m, i \in [0, l^w - 1]$. Then, d_i^w represents the edge between 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, $d_i^w \in \{-1, 1\}$, $i \in [1, l^w]$. Besides, l^w denotes the route's length from the root to the leaf word.

Using this schema, we can change the traditional word index or 1-of-K representation into a word polarity encoding (d^w, θ^w) for word w by denoting the polarity route to locate each word.

3.2 Cost Function Compression

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

$$p(d_i^w = \pm 1|\theta_{i-1}^w, h) = \sigma(d_i^w \theta_{i-1}^w h), \tag{5}$$

where the symmetric rule for a sigmoid function: $\sigma(z) + \sigma(-z) = 1$, is employed for abbreviation [Minka, 2003].

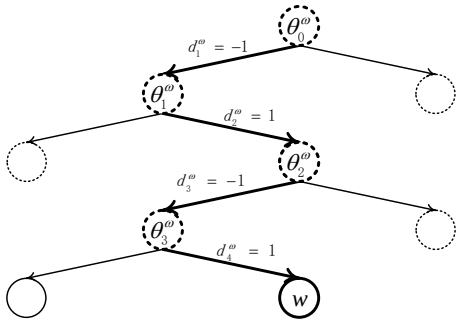

Figure 2: Tree-based Hierarchical Softmax. Dotted circles denote internal nodes parameterized by θ_i^w and edges between nodes are parameterized by d_i^w. In addition, the full circle denotes the leaf word w.

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:

$$\begin{aligned} p(w|h) = \prod_{i=1}^{l^w} p(d_i^w, \theta_{i-1}^w|h) = \prod_{i=1}^{l^w} \sigma(d_i^w \theta_{i-1}^w h). \\ = \sigma({d^w}^\top \theta^w h) \end{aligned} \tag{6}$$

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

$$\begin{aligned} \mathcal{L}(\theta|h, w) &= \sum_{i=1}^{l^w} \log(1 + \exp(-d_i^w \theta_{i-1}^w h)) \\ &= \sum_{i=1}^{l^w} \text{softplus}(-d_i^w \theta_{i-1}^w h), \\ &= \text{softplus}(-{d^w}^\top \theta^w h) \end{aligned} \tag{7}$$

where the softplus function is defined as: $f(z) = \log(1 + \exp(z))$, which shows that the p-tHSM loss belongs to the spherical family [Vincent *et al.*, 2015; de Brébisson and Vincent, 2016].

In order to optimise the model's parameters, we need to calculate the gradient as follows:

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

A major advantage of decomposing the multi-label prediction problem into binary tree classification is that it avoids normalizing the probability over the whole vocabulary, as the sum of all the probabilities for words in the hierarchical word tree is naturally equal 1.

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

In the inference procedure, for a node in the i-th layer, we choose the left branch when $p(d_i^w|\theta_{i-1}^w, h) \geq 0.5$ and select

the right child when the opposite applies. Thus, the computational time complexity is $\mathcal{O}(\log \mathcal{V})$.

During implementation, we maintain two large matrices D and Θ, and (d^w, θ^w) is retrieved by obtaining the w-th row vector separately, where w is the index in the vocabulary. Furthermore, (d^w, θ^w) is defined by the word hierarchy criterion, which is introduced in the following section, and θ^w is optimized based on the training dataset.

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 tHSM, as follows.

Word2vec via kmeans: 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 denoting the lower bound of the tree modelling method.

Frequency Binning: This method sorts the vocabulary based on word frequency and can directed applied on tHSM model. Besides, words can be partitioned into euqal size classes thereby obtaining clusters of words with similar frequency [Mikolov *et al.*, 2013b].

Brown Clustering: The Brown algorithm is an agglomerative hierarchical clustering algorithm that clusters words to maximize the mutual information for bigram words [Brown *et al.*, 1992; Liang, 2005], This is a class-based bigram language model, which can generate word binary prefix string.

Clark clustering: 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.

4 Experimental Study

4.1 Recurrent Language Model

The goal of LM 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:

$$p(w_1, w_2 \cdots, w_T) = \prod_{t=0}^{T} p(w_t | w_{<t}), \tag{10}$$

where $p(w_t|w_{<t})$ denotes the probability of the next-step word given its previous context $[w_1, \cdots, w_{t-1}]$ as inputs and is usually modeled by RNN model, which utilize time-series information of sequences.

In this research, recurrent neural network with gated units (GRU) [Chung *et al.*, 2014] is preferred because it requires a smaller set of parameters to model the internal recurrence mechanism at the same time its gating function simplifies the gradient vanishing problem [Hochreiter, 1998]. Hence, the applied GRU units is given as follows:

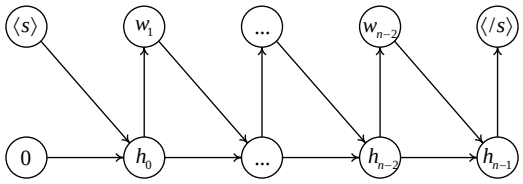

Figure 3: Recurrent Neural Language Model. Every sentence is wrapped with start (i.e., $\langle s \rangle$) and end (i.e., $\langle$ /s$\rangle$) tokens.

$$\begin{aligned} z_t &= \sigma(W^z[h_{t-1}, x_t] + b^z) \\ r_t &= \sigma(W^r[h_{t-1}, x_t] + b^r) \\ g_t &= \phi(W^h[r_t \odot h_{t-1}, x_t] + b^h) \\ h_t &= z_t \odot g_t + (1 - z_t) \odot h_{t-1}, \end{aligned} \tag{11}$$

where σ and ϕ are element-wise squashing functions that force the gating values in $[0, 1]$ and $[-1, 1]$, respectively. We fix σ as sigmoid and ϕ as a hyperbolic tangent function. In addition, $\odot$ denotes element-wise matrix multiplication.

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:

$$\mathcal{L}(\theta | w_1, \cdots, w_T) = -\frac{1}{T} \sum_{t}^{T} \log p(w_t | w_{<t}) \tag{12}$$

where T is the number of all the words in the test case.

4.2 Experiment Setups

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. A lower perplexity for a language model implies better predictability.

Perplexity is an intrinsic evaluation metric for a model and it increases exponentially relative to the model's training loss function: $\mathcal{NLL} = \log \mathcal{PP}(w_1, \cdots, w_T)$, denoting that we are directly optimizing the *perplexity* metric during training.

$$\mathcal{PP}(w_1, \cdots, w_T) = \sqrt[T]{\frac{1}{\prod_{t=0}^{T} p(w_t | w_{<t})}} \tag{13}$$

In addition, the training efficiency should be considered as a large model will consume more time during training. Thus, we analysed the training and inference time complexity for different optimizations in both theoretical and empirical terms.

We conducted experiments on the standard Penn Tree Bank (PTB) [Marcus *et al.*, 1993], WikiText-2 and WikiText-103 dataset [Merity *et al.*, 2016] to evaluate their accuracy and efficiency, and the detailed statistics [1] is shown in Table 1.

[1] 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
Words	887,521	70,390	78,669	2,088,628	217,646	245,569	103,227,021	217,646	245,569
#Vocab Size		10,000			33,278			267,735	
OOV (%)		4.8%			2.6%			0.4%	

4.3 Results and Discussions

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 stabilize the performance of p-tHSM.

Table 2: Perplexity Using cHSM and p-tHSM with various Clustering Methods on PTB Dataset.

Methods	p-tHSM	cHSM
Word2vec	184	175
Freqs Binning	127	152
Clark Cluster	155	168
Brown Cluster	**121**	**131**

Next, we compared several tree clustering criteria using the PTB dataset, as shown in Table 2. Randomly shuffling words and splitting them into clusters of equal sizes performed the worst because it did not provide any information about the distribution of the words. The frequency binning method categorizes words into frequency groups, where words in the same branches of the tree share a similar frequency [Mikolov *et al.*, 2013a]. This method performed better than alphabetical order clustering, which sorts words according to their alphabetical order and splits them into groups of words. Brown clustering takes time to calculate the bigram of texts and similar words are grouped together in the feature space while it performs the best.

Finally, p-tHSM performed relatively better than the cHSM model, except when we used the random shuffling method. This indicates that deeper tree models are more suitable for word clustering and that inappropriate clustering can reduce the efficiency of the tree hierarchy.

Factored Output Layer Comparison

We benchmarked the time complexity for training the same batched data empirically and theoretically, and the detailed results are shown in Table 3, where the "Softmax" method denotes the traditional flattened softmax without any optimization. We tried to process the PTB dataset with these algorithms and we calculated the average time required for processing one batch data. In addition, the input word-embedding, hidden layer, and output vocabulary were set as $\{100, 250, 10000\}$, respectively, and the batch size was fixed to 50 [Chen *et al.*, 2016].

Table 3 shows the time consumption required from the input of the data until the model's cost was calculated, which is denoted as the "Forward" process. The "Backward" process denotes the process for the error propagated from the output layer to the input layer. Clearly, the p-tHSM performed relatively better than the baseline methods in both the practical and theoretical evaluations.

Table 3: Factored Output Layer Time Comparison.

Method	Complexity	Forward (ms)	Backward (ms)
Softmax	$\mathcal{O}(\mathcal{V})$	118.30	3758.93
cHSM	$\mathcal{O}(\sqrt{\mathcal{V}})$	45.68	220.41
tHSM	$\mathcal{O}(\log \mathcal{V})$	40.38	181.52
p-tHSM	$\mathcal{O}(\mathcal{C})$	**33.07**	**64.62**

Furthermore, we evaluated the scalability of the p-tHSM models relative to the vocabulary size and the detailed results are shown in Fig. 4. In order to visualize the impact of the p-tHSM algorithm, the "Softmax" method is not included because it consumed more computational time compared with the others. Clearly, the cHSM scales with the square root of the vocabulary size (i.e., $\sqrt{\mathcal{V}}$) whereas p-tHSM exhibits stationary performance as the vocabulary size increases.

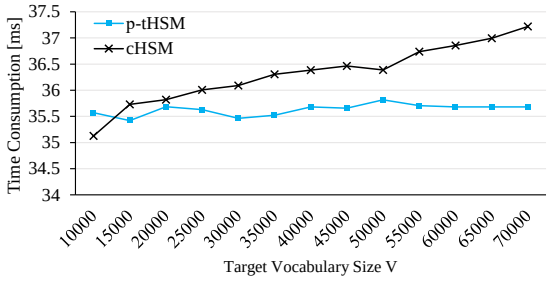

Figure 4: Scalability of cHSM and p-tHSM Relative to the Vocabulary Size.

Based on these experiments, we may conclude that the proposed p-tHSM method can solve the word probability on a large vocabulary with constant time complexity $\mathcal{O}(\mathcal{C})$. In particular, the time required by p-tHSM does not scale with the size of the target vocabulary, thereby indicating that we achieved a satisfactory speed-up ratio for the hierarchical softmax architecture and it outperformed the historical benchmark $\mathcal{O}(\log \mathcal{V})$. This performance is attributable to the acceleration based on the hardware's parallelism, but it also due

Table 4: Perplexity Summarisation On PTB, WikiText-2 and WikiText-103 Dataset

Methods	PTB	WikiText-2	WikiText-103
GRU + Softmax	166.90	330.95	301.76
GRU + NCE [Gutmann and Hyvärinen, 2010]	134.35	275.15	254.64
GRU + Blackout [Ji *et al.*, 2016]	122.49	264.31	234.76
GRU + cHSM [Chen *et al.*, 2016]	124.05	243.44	213.38
GRU + p-tHSM (pretrained by Huffman [Mikolov *et al.*, 2013b])	127.34	231.24	202.25
GRU + p-tHSM (pretrained by Brown [Brown *et al.*, 1992])	**121.11**	**220.40**	**195.57**

to the fundamental structure of the *word polarity encoding* scheme, which can maintain the target word tree efficiently and in a parallel manner.

Sampling-based Approximation Comparison

The efficiency and accuracy of sampling-based algorithms are strongly related to the sample size, which we tested in the next evaluation. We tested several sample sizes to assess their effects on the blackout and NCE approximations, and the results are shown in Table 5. According to Table 5, the blackout algorithm performed relatively better than the traditional NCE model, which is consistent with the results of the experiments reported [Ji *et al.*, 2016] .

Table 5: Perplexity Comparision of Different Sampling Sizes on the NCE and Blackout Algorithms on PTB Dataset.

Sample size	NCE	Blackout
1	359	321
100	277	214
1000	159	147
10000	134	122

We found that a sample size of one and binary classification performed the worst. The training process involves learning in the true word probability phase as well as learning in the noise probability estimation phase. Thus, these algorithms can only estimate the noise probability accurately when sufficient noise samples have been collected.

The performance of these estimation algorithms converged to optimum perplexity as the number of noise samples increased, but the speed-up ratio V/k decreased, where k denotes the sample size. Thus, we regard this term as a hyperparameter and it should be tuned based on the validation set, so the sample size was not fixed during training. The sampling method cannot be applied during inference and the original softmax method is employed instead.

All Experiments Benchmark

As shown in Table 4, all experiments implemented with Theano framework [Al-Rfou *et al.*, 2016] were run on one standalone GPU device with 12 GB of graphical memory (i.e., Nvidia K40), which allowed the large matrix multiplications to be possible. However, the resource of one GPU memory are exploited quickly with the rise of parameters.

After replacing the traditional huffman coding scheme in tHSM with a word polarity coding scheme and a compact softplus form cost function, 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 from $\log V$ to $\mathcal{O}(\mathcal{V})$ and obtained the optimum speed-up ratio for large vocabulary problems. Furthermore, after testing several existing tree clustering algorithms for stabilizing 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 language models. In 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.

The first approach was rejected because it can not generalise to large real world tasks. The second type of method can reduce the training time consumption in an efficient manner by not summarizing all the words through sampling, but it fails during inference because a prior distribution is applied. The third method changes the flattens architecture of the softmax algorithm into a heuristic tree structure with two possible types: cHSM and tHSM. The cHSM method builds on the two-step softmax classification method and tHSM extends this idea to $\log \mathcal{V}$ steps binary classification tasks.

We built a compact mathematical model for tHSM's variants and exploited the advantages of GPU hardware and the proposed encoding scheme to improve the standard tHSM, so it can efficiently calculate the exact softmax and gradient with a large target vocabulary. Remarkably, the complexity of this algorithm is independent of the vocabulary size and it can be computed with constant time complexity, thereby allowing it to tackle very large vocabulary problems. Furthermore, we evaluated several word 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 original softmax layer and more efficient tree clustering was obtained, so the performance was better compared with other sampling-based optimizations.

In future research, we plan to explore the application of the softplus function and build a connection with the Z-loss function family [de Brébisson and Vincent, 2016]. In addition, several clustering algorithms are applied with huge time consumption, which can be optimised in order to design a more efficient hierarchical structure.

References

[Al-Rfou *et al.*, 2016] Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, and et al. Theano: A python framework for fast computation of mathematical expressions. *CoRR*, abs/1605.02688, 2016.

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

[Bengio and Senecal, 2008] Yoshua Bengio and Jean-Sébastien Senecal. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks*, 19(4):713–722, 2008.

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

[Brown *et al.*, 1992] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479, 1992.

[Chen *et al.*, 2014] Xie Chen, Yongqiang Wang, Xunying Liu, Mark J. F. Gales, and Philip C. Woodland. Efficient gpu-based training of recurrent neural network language models using spliced sentence bunch. In *Interspeech*, pages 641–645, 2014.

[Chen *et al.*, 2016] Wenlin Chen, David Grangier, and Michael Auli. Strategies for training large vocabulary neural language models. In *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.

[de Brébisson and Vincent, 2016] Alexandre de Brébisson and Pascal Vincent. The z-loss: a shift and scale invariant classification loss belonging to the spherical family. 2016.

[Derczynski *et al.*, 2015] Leon Derczynski, Sean Chester, and Kenneth S. Bøgh. Tune your brown clustering, please. In *RANLP*, pages 110–117, 2015.

[Grave *et al.*, 2016] Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. Efficient softmax approximation for gpus. *CoRR*, abs/1609.04309, 2016.

[Gutmann and Hyvärinen, 2010] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *JMLR*, 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. *JMLR*, 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 *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. *ICLR*, 2016.

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

[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. *COLING*, 19(2):313–330, 1993.

[Merity *et al.*, 2016] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *CoRR*, abs/1609.07843, 2016.

[Mikolov *et al.*, 2010] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *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 *NIPS*, pages 3111–3119. 2013.

[Minka, 2003] Thomas P Minka. Algorithms for maximum-likelihood logistic regression. 2003.

[Mnih and Kavukcuoglu, 2013] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *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 *ICML*, 2012.

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

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

[Vincent *et al.*, 2015] Pascal Vincent, Alexandre de Brébisson, and Xavier Bouthillier. Efficient exact gradient update for training deep networks with very large sparse targets. In *NIPS*, pages 1108–1116. 2015.

[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.