Learning with Noise-Contrastive Estimation: Easing training by learning to scale

Matthieu Labeau, Alexandre Allauzen

Learning with Noise-Contrastive Estimation: Easing training by learning to scale.

Matthieu Labeau and **Alexandre Allauzen**
LIMSI, CNRS, Univ. Paris-Sud, Université Paris-Saclay, 91405 Orsay, France
{matthieu.labeau,alexandre.allauzen}@limsi.fr

Abstract

Noise-Contrastive Estimation (NCE) is a learning criterion that is regularly used to train neural language models in place of Maximum Likelihood Estimation, since it avoids the computational bottleneck caused by the output softmax. In this paper, we analyse and explain some of the weaknesses of this objective function, linked to the mechanism of *self-normalization*, by closely monitoring comparative experiments. We then explore several remedies and modifications to propose tractable and efficient NCE training strategies. In particular, we propose to make the scaling factor a trainable parameter of the model, and to use the noise distribution to initialize the output bias. These solutions, yet simple, yield stable and competitive performances in either small and large scale language modelling tasks.

1 Introduction

In many tasks, such as machine translation and speech recognition, statistical language models[1] play a key role. Neural models (Bengio et al., 2003; Mikolov et al., 2010; Józefowicz et al., 2016) have recently shown great improvement. However most of them share a common issue: a large output vocabulary implies a prohibitive computation time, due to the output normalization, along with a prediction challenge in a such high dimensional space. A workaround is to reduce the vocabulary size by considering sub-word units like morphemes or even characters. For several applications, this is an efficient solution, though the main issue is not directly addressed. Other trends consist in changing the output structure by using *shortlists* (Schwenk, 2007) or *hierarchical softmax* (Morin and Bengio, 2005; Mnih and Hinton, 2009; Le et al., 2011), while *self-normalisation* techniques (Devlin et al., 2014; Andreas et al., 2015; Chen et al., 2016) partially solve the issue at test time. Finally, sampling-based techniques like *Importance sampling* (Bengio and Sénécal, 2003; Jean et al., 2015), and *Noise-Contrastive Estimation* or NCE (Mnih and Teh, 2012) are also promising alternatives.

In this work, we focus on NCE, which uses a discriminative objective that approximates negative log-likelihood. Its main advantage is to consider the model as *un-normalized* instead of trying to approximate its normalization. With this objective function the model is explicitly learnt to estimate *un-normalized* probability distributions, while the partition function (or the scaling factor) is parametrized separately. While this method is very appealling in theory, empirical issues arise in large vocabulary applications: training divergence and instability, along with poor performance when compared to similar approaches, notably Importance Sampling. The contributions of this paper are twofold: first an empirical exploration of the NCE allows us to better explain the estimation process and why it sometimes fails; then we propose and explore several remedies yielding to tractable and efficient NCE training strategies.

While section 2 reviews two widely used training criteria for un-normalized model, Importance Sampling and NCE, section 3 provides an empirical analysis of the NCE. This algorithm is theoretically proven to converge when the partition function is parametrized separately. However, the scaling parameter

[1]Neural Machine Translation and Speech Recognition can be seen as a language model where the sequence probability is conditioned on a source sentence or a speech signal.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3090–3101
Santa Fe, New Mexico, USA, August 20-26, 2018.

is usually fixed and the *un-normalized* model therefore *self-normalizes*, as a side effect of the training procedure. We show experimentally that the training instability is tied to the difficulty of the model to self-normalize. After analysing the consequences of these issues, we experiment (section 4) with various solutions, including smoothing the sampling distribution or using the recent application of *Negative sampling* to language modelling (Melamud et al., 2017). Moreover, we propose to jointly learn the scaling parameter with the model and show that this approach yields a practical and efficient training strategy. We also propose to initialize the output bias to the logarithm of the noise distribution, which diminishes greatly the impact of the issues described in section 3.

2 Training objectives and partition function

For neural language models (NLMs), the main computational burden lies in the summations over the (very large) vocabulary $\mathcal{V}$. Indeed, a NLM parametrized by θ, outputs, for an input context H, a conditional distribution P_θ^H for the next word, over $\mathcal{V}$. This conditional distribution is defined using the *softmax* activation function:

$$P_\theta(w|H) = \frac{e^{s_\theta(w,H)}}{\sum\limits_{w' \in \mathcal{V}} e^{s_\theta(w',H)}} = \frac{e^{s_\theta(w,H)}}{Z_\theta(H)} \tag{1}$$

Here, $s_\theta(w, H)$ is a scoring function which depends on the network architecture. The denominator is the *partition function* $Z_\theta(H)$, which is used to ensure that for each input context H, output scores are normalized into a probability distribution. Then, each model can be written as an *un-normalized* model divided by the partition function. The natural objective is to minimize the negative log-likelihood of this conditional distribution for each tuple of input context and following word $(H, w) \in \mathcal{D}$, where $\mathcal{D}$ is the training set:

$$NLL(\theta) = -\sum_{(H,w)\in\mathcal{D}} \log P_\theta(w|H) \tag{2}$$

Using the Stochastic Gradient Descent (SGD) to train this objective implies taking the objectives gradient to make the parameter updates. For one training example (H, w), the gradient of the log-probability will be computed as follows:

$$\frac{\partial}{\partial\theta} \log P_\theta(w|H) = \frac{\partial}{\partial\theta} s_\theta(w, H) - \sum_{w' \in \mathcal{V}} P_\theta(w'|H) \frac{\partial}{\partial\theta} s_\theta(w', H). \tag{3}$$

The first term tends to increase the conditional log-likelihood of the word w, whereas the second term lowers probabilities for all the other words in the vocabulary. Unfortunately, the partition function $Z_\theta(H)$ is necessary to compute both $P_\theta(w'|H)$ at test time, and the parameter gradients are necessary to update the model during the training process. In (Gutmann and Hyvärinen, 2013), the authors detail why knowing the partition function, which represents the *'scale'* of the model, is essential to compute the likelihood[2]. Parametrizing separately the partition function (as a scaling parameter) would give irrelevant results, since we could minimize the negative log-likelihood independently of the data. Following this reasoning, we focus on objectives that allow for the estimation of un-normalized models. In the language modelling literature, methods based on Importance Sampling (IS) and NCE are the most widely used.

2.1 IS: approximating the partition function

As detailed in (Bengio and Sénécal, 2003), the idea is to rewrite the gradient described in equation 3 as an expectation that we estimate using importance sampling. We choose a distribution P_n from which it is easy to sample, and obtain the following gradient approximation:

[2]Beyond the fact that the concept of likelihood only applies to probability density functions, which are normalized.

$$\frac{\partial}{\partial\theta}\log P_\theta(w|H) \approx \frac{\partial}{\partial\theta}s_\theta(w,H) - \frac{1}{k}\frac{1}{Z_\theta(H)}\sum_{\substack{i=1\\ \hat{w}_i\sim P_n}}^{k}\frac{e^{s_\theta(\hat{w}_i,H)}}{P_n(\hat{w}_i)}\frac{\partial}{\partial\theta}s_\theta(\hat{w}_i,H) \tag{4}$$

While we replaced one summation over $\mathcal{V}$, the partition function remains. Rewriting it as an expectation, we can apply importance sampling a second time, using the same distribution P_n:

$$Z_\theta(H) = \mathbb{E}[e^{s_\theta(w,H)}] \approx \frac{1}{k}\sum_{\substack{i=1\\ \hat{w}_i\sim P_n}}^{k}\frac{e^{s_\theta(\hat{w}_i,H)}}{P_n(\hat{w}_i)} \tag{5}$$

And thus, by approximating the partition function, we obtain a biased estimator for the maximum likelihood gradient. Both its bias and variance can be reduced by increasing the sample size k. In order to limit this growth, (Bengio and Sénécal, 2008) investigates on adapting P_n during training.

2.2 NCE: avoiding normalization

NCE was first described in (Gutmann and Hyvärinen, 2010), as a way of estimating a parametric probabilistic model from data in the case where the probability function of the model is un-normalized. Considering the partition function as a separate parameter, the objective function mimics maximum-likelihood estimation by learning to discriminate between true examples from data or generated from a *noise distribution*. This method has been applied to language modelling in (Mnih and Teh, 2012): considering a mixture of the data and noise distributions, for each example $(H, w) \in \mathcal{D}$, we draw k noise samples from a noise distribution P_n. The posterior probability of the class C associated to the sample can be estimated ($C = 1$ if the sample comes for the data and $C = 0$ from the noise). Since the goal is to approximate the data distribution with a model parametrized by θ, the conditional class distribution is defined as $P(w|C = 1, H) = P_\theta(w|H)$ and $P(w|C = 0, H) = P_n(w)$, which gives the posterior class probabilities:

$$P(C = 1|w, H) = \frac{P_\theta(w|H)}{P_\theta(w|H) + kP_n(w)} \quad \text{and} \quad P(C = 0|w, H) = \frac{kP_n(w)}{P_\theta(w|H) + kP_n(w)} \tag{6}$$

If σ denotes the sigmoid function, these equations can be rewritten as a logistic regression problem:

$$P(C = 1|w, H) = \sigma\left(\log\frac{1}{k}\frac{P_\theta(w|H)}{P_n(w)}\right) \tag{7}$$

The reformulation obtained in equation 7 shows that training a classifier based on a logistic regression will estimate the log-ratio of two distributions: this allows the learned distribution to be un-normalized, and the partition function to be parametrized separately[3]. However, the partition function is context-dependent. In (Mnih and Teh, 2012), the authors argue that these context-dependent scaling parameters can be put to one, and that given the number of free parameters, the output scores $e^{s_\theta(w,H)}$ will self-normalize for each context. In what follows, we adopt this trick and for clarity denote $p_\theta(w|H) = e^{s_\theta(w,H)}$ the un-normalized model score. Since the class labels are by assumption Bernoulli distributed and independent, the classification objective is given by maximizing the log-likelihood of the true examples to belong to class $C = 1$ and the noise samples $(\hat{w}_i)_{1\leq i\leq k}$ to $C = 0$, which is, for one example (H, w) from $\mathcal{D}$:

$$J_\theta^H(w) = \log\frac{p_\theta(w|H)}{p_\theta(w|H) + kP_n(w)} + \sum_{i=1}^{k}\log\frac{kP_n(\hat{w}_i)}{p_\theta(\hat{w}_i|H) + kP_n(\hat{w}_i)} \tag{8}$$

The gradient update is the following:

[3] (Pihlaja et al., 2012) and (Gutmann and Hirayama, 2011) show the NCE to be a particular case of larger classes of objective functions, with which the model learns to match a ratio of data and noise samples instead of directly matching the data samples, which allows un-normalized model estimation

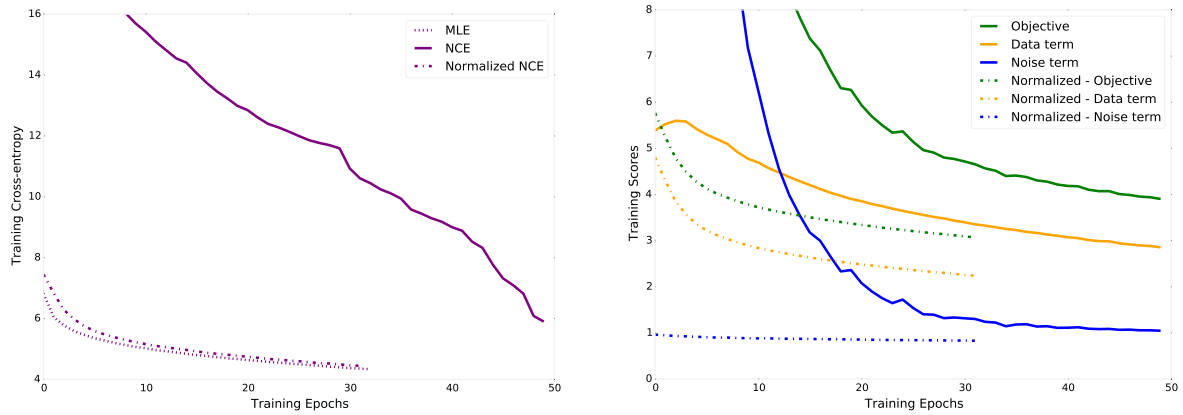

Figure 1: *Left*: Training cross-entropy curves on PTB with Maximum-Likelihood estimation, NCE on an un-normalized model and a model normalized before application of the NCE. *Right*: Training scores (see equation 10) of the same un-normalized and normalized models trained with NCE

$$\frac{\partial}{\partial\theta}J_\theta^H(w) = \frac{kP_n(w)}{p_\theta(w|H) + kP_n(w)}\frac{\partial}{\partial\theta}\log p_\theta(w|H) - \sum_{i=1}^{k}\left[\frac{p_\theta(\hat{w}_i|H)}{p_\theta(\hat{w}_i|H) + kP_n(\hat{w}_i)}\frac{\partial}{\partial\theta}\log p_\theta(\hat{w}_i|H)\right] \tag{9}$$

It is worth noting that this gradient converges to the Maximum-Likelihood Estimation (MLE) gradient as the number of samples k grows. Concerning the choice of the noise distribution, the estimation error of parameters θ is asymptotically independent of P_n when the ratio of noise samples by example k coming from the data is large enough. It is also shown that having both a noise distribution close to the data distribution and a high number of samples k lower the estimation error. (Mnih and Teh, 2012) compared using the uniform and unigram distribution in their experiments, finding the unigram distribution to give far more accurate results. In the literature, NCE was then mainly used in the context of machine translation: (Vaswani et al., 2013; Baltescu and Blunsom, 2015) report results with the unigram distribution, while (Zoph et al., 2016) used an uniform noise. However, despite strong theoretical guarantees, (Chen et al., 2016) highlighted the inconsistency of NCE training when dealing with very large vocabularies, showing very different perplexity results for close loss values. In another work (Józefowicz et al., 2016), NCE was shown far less data-efficient than IS.

3 Training behaviour of NCE

To understand these instabilities, the training process of the same language model is monitored, varying only the training criterion, *i.e* MLE and NCE. For a better understanding, we also consider an '*intermediate*' model denoted as *NCE normalized*. This model is trained using NCE; however, we normalize the scores $p_\theta(w|H) = e^{s_\theta(w,H)}$ into $P_\theta(w|H)$ right before computing the objective. While it is without any practical interest, this model will allow us to better assess the impact of the normalization process. We train the 3 models on the Penn Tree Bank (PTB) with a full vocabulary and $k = 100$ noise samples drawn from the unigram distribution for NCE. We use classic SGD[4]. To reduce the impact of the training criterion, models are learnt for a minimum of 30 epochs. Beyond that limit, we backtrack the epoch when no progress has been made on the validation set perplexity, stopping training after 10 consecutive backtrackings[5].

3.1 The objective and partition functions

The training cross-entropy for the 3 models are drawn in the top graph of figure 1. When trained with MLE or NCE, the normalized model exhibits similar learning curves. However, the un-normalized model

[4]Hyperparameters are the same than those used in (Melamud et al., 2017), except the vocabulary, i.e $\approx 44K$ words, instead of $10K$, to consider a more realistic learning situation for NCE.

[5]For the sake of clarity, backtrakings are discarded from the graphs, keeping only the epochs used to obtain the final model.

trained with NCE takes far longer to reach a comparable cross-entropy, ending with a higher value. In the bottom graph we can observe, for the normalized and un-normalized models trained with NCE, the values of minus the objective function[6], and both of its terms (the first, data-dependent, and the second, containing the noise samples):

$$-J_\theta = \sum_{H,w\in\mathcal{D}} -\log \frac{p_\theta(w|H)}{p_\theta(w|H)+kP_n(w)} - \sum_{H,w\in\mathcal{D}}\sum_{i=1}^{k} \log \frac{kP_n(\hat{w}_i)}{p_\theta(\hat{w}_i|H)+kP_n(\hat{w}_i)} \tag{10}$$

We can observe a small decrease of the second term for the normalized model, whereas for the un-normalized, it starts with high values and decreases to become closer to the normalized one. Moreover, the gap between the two data-dependent terms stays high at the end of training. For a deeper analysis, we study the values of the partition function during training. For both the normalized and the un-normalized models, the partition function $Z_\theta(H)$ associated to each training context H is computed and the results over an epoch are summarized with an histogram. Each bin represents a range of values, and its height represent the fraction of training contexts whose partition function belongs to this range. For a better readability, both range values and bin heights are in log-scale. These histograms are shown in figure 2 for different epochs. While the repartition of the partition function values for the normalized model does not change much - which is logical, since the model is always integrated to 1 - we observe that these values for the un-normalized model are chunked together and are decreasing during learning (epochs 5, 10 and 15). However, this trend seems to slow down towards the end of training, since, at epoch 30, the repartition resemble to the one of the normalized model, only shifted to smaller values (which are still quite high).

3.2 Self-normalisation is crucial for NCE

Considering these results, we dissect what happens during the training of the un-normalized model. Let us rewrite the objective J_θ as function of the following ratio:

$$r_\theta(w|H) = \frac{p_\theta(w|H)}{kP_n(w)}, \text{ then } -J_\theta = \sum_{H,w\in\mathcal{D}} -\log \frac{r_\theta(w|H)}{r_\theta(w|H)+1} - \sum_{H,w\in\mathcal{D}}\sum_{i=1}^{k} \log \frac{1}{r_\theta(\hat{w}_i|H)+1} \tag{11}$$

[6]Since the objective J_θ^H is negative, we minimize $-J_\theta^H$.

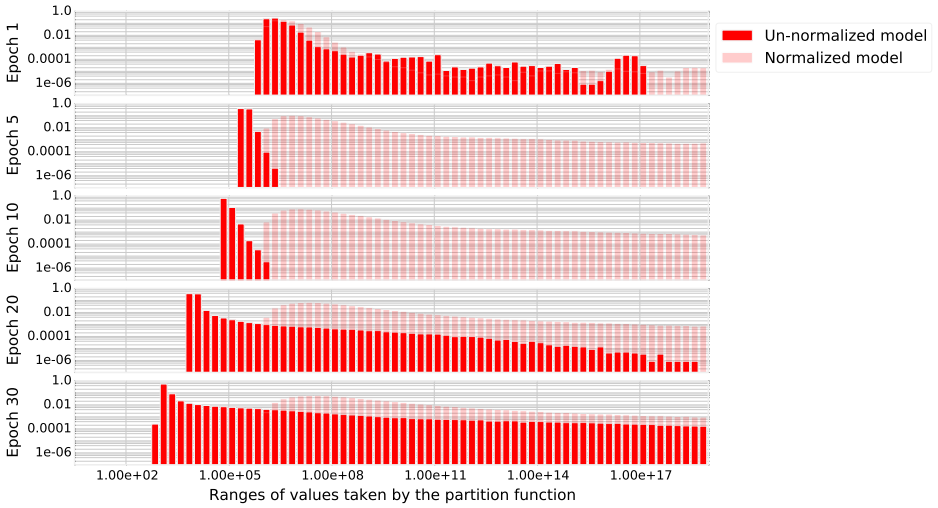

Figure 2: Repartition of the values of the partition function $Z_\theta(H)$ for all examples (H, w) during specific epochs of training on PTB with NCE. The fully coloured bars represent the repartition for the un-normalized model, while the faded bars represent the repartition for the normalized model. Both scales are logarithmic.

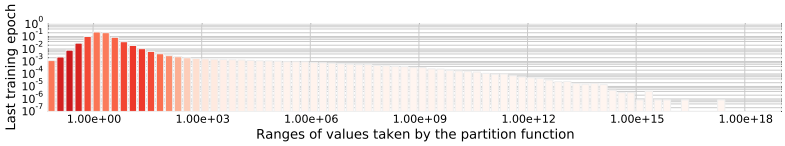

Figure 3: Repartition of the values of the partition function $Z_\theta(H)$ for all training examples (H, w), during the last epoch of training with NCE on PTB. For both this figure and figure 4, the colour of a bin indicates the proportion of training examples (H, w) for which the word w is one of the 10 most frequent according to the noise distribution P_n: the lighter the colour is, the higher is that proportion. Both scales are logarithmic.

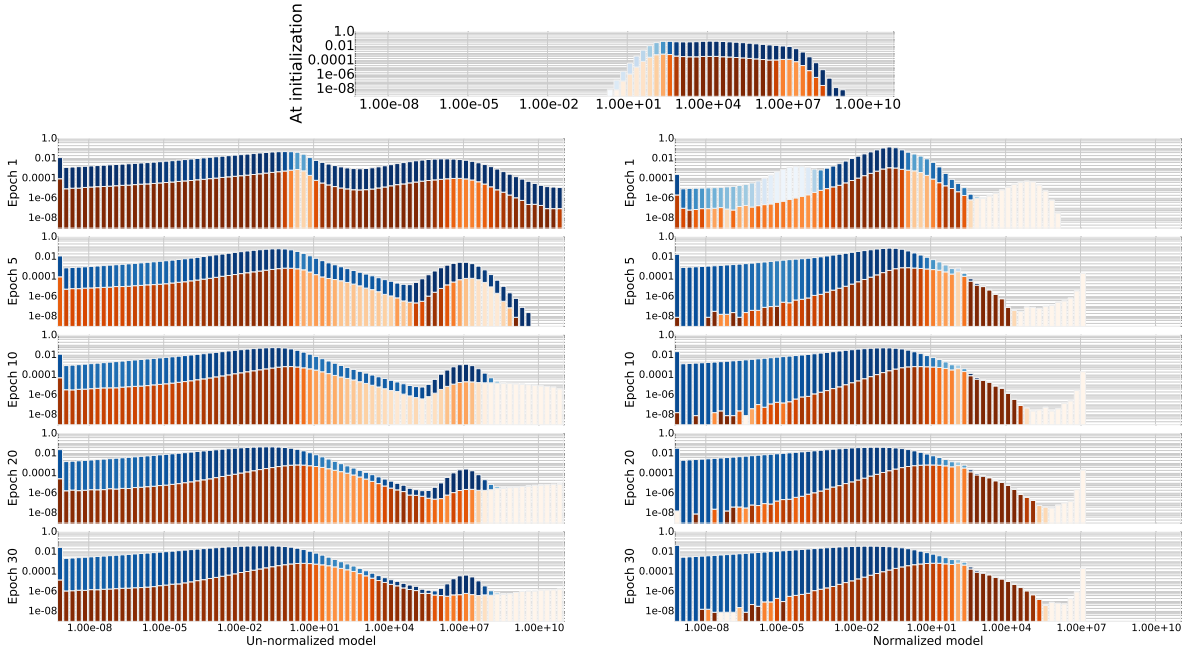

Figure 4: Repartition of the values of the ratio $r_\theta(w|H)$ for all positive training examples (H, w) in orange, and associated noise sample $(\hat{w}_i)_{i=1}^k$, in blue, at initialization (top) and during several training epochs with NCE on PTB. On the left, the model is un-normalized, while on the right it is normalized.

The second term of $-J_\theta$ is very high for large values of r_θ. Since values of P_n are always smaller than 1 and $k = 100$, and knowing the partition function at the beginning of training is large for all contexts, we can assume that $r_\theta(\hat{w}_i|H)$ will be large for a least part of the set of noise samples $(\hat{w}_i)_{i=1}^k$. Thus, this second term is the largest part of the objective, whose minimization leads to a decrease of the model scale. However, this is done at the expense of the data-dependent term, since its score increases a little during the first few epochs. If we look at its gradient, we see from equation 9 that it is always scaled by the weight $1/(r_\theta(w|H) + 1)$, which means that the objective will learn nearly nothing from data as long as the model scale has not decreased to a reasonable value. This shows that this 'self-normalization' mechanism is crucial for NCE, but we still need to understand what makes a scaling value 'reasonable'.

3.3 Impact of the noise distribution P_n

While the un-normalized model underperforms its normalized counterpart, its cross-entropy is not that far off. Let us explore the repartition of the partition function values during the last epoch of training. In particular, we expect an impact of the P_n values. The histogram is shown in figure 3: each bin is coloured according to the proportion of examples associated to the highest values of P_n it contains[7]. We clearly observe that while the majority of the partition functions have values under ≈ 100, functions associated

[7]Since we use the unigram distribution, it indicates here the examples who are among the 10 most frequent words.

with the words with the higher values of P_n - which are also the most sampled words - can take values that are far larger. To understand why, consider the weight of the gradient of the data-dependent term:

$$\frac{1}{r_\theta(w|H)+1} = \frac{kP_n(w)}{p_\theta(w|H)+kP_n(w)}.$$

Having higher values of P_n means that the model does not need to scale back the partition function as much to be able to learn from data. Besides, if the noise distribution is correlated with the data distribution, an example with a high value of P_n tends to be more frequent. Therefore the model will be able to learn first from words w with the higher values of $P_n(w)$. However, the lower $P_n(w)$ is, the more difficult it will be for the model to learn about w. And the associated partition function will have to be closer to 1 for the data-term gradient to be meaningful. To visualize how learning is affected, we study histograms showing the repartition of the values of r_θ during a training epoch. They are presented in figure 4. Again, we show histograms for both models, at epochs 1, 5, 10, 20, and 30, while on top is a histogram of the initial repartition (before learning). Similarly, they are in log-scale. On the right are histograms for the normalized model. For epoch 1, there is a peak of data examples with higher values of P_n - frequent words - that have higher values of r_θ, while for noise samples, the repartition still follows the initial one - samples with higher P_n have a smaller r_θ. As learning progresses, a clear trend appears: values of r_θ for data examples, seemingly ordered by values of P_n, become well-separated of the values of r_θ for noise samples. On the left are the repartition of values for the un-normalized model. While they are ordered by values of P_n, a clear separation between data examples values and noise samples values takes far more time to appear, and a cluster of high values of r_θ for noise samples still remains visible at epoch 30 - these high values correspond to small values of P_n: in our case, rare words.

These observations show us that an higher discrepancy between high and low P_n leads to a more difficult learning procedure with NCE. As a consequence, and because of the Zipfian distribution of word frequencies, using the unigram distribution will be harder as the size of the vocabulary grows.

4 Practical solutions and learning to scale

As discussed in section 3, two quantities strongly impact NCE training: first, the noise distribution (and the number k of noise samples); then, the partition function. To evaluate different training strategies, we provide training cross-entropy curves in figure 5. As a comparison point, results with importance sampling are also reported since it is nowadays the best choice to train large language models. To verify the validity of our analysis on the PTB, we also report results on a larger corpus, the 1 Billion Word Benchmark[8] (Chelba et al., 2014). In this case, the vocabulary contains $64K$ words, which renders normalized models not competitive[9]. We trained the models for at least one epoch, and up to a second if models made progress on the validation set perplexity. Final perplexity results are presented in table 1.

4.1 Acting on the noise distribution P_n

As the number of noise samples k increases, the estimation error reduces, along with the necessity for the model to downscale the partition function. With $k = 500$, we can observe indeed a faster convergence with a lower final perplexity. On the large corpus, the conventional NCE model doesn't even reach a perplexity under the size of the vocabulary, whereas augmenting k to 500 is not enough to reach a suitable perplexity. However, increasing k reduces the computational benefit of NCE, even with k far inferior to the vocabulary size. There is a trade-off in the choice of P_n: the closer P_n is to the data distribution, the lower is the estimation error; however the Zipfian distribution of word frequencies implies too few updates for rare words if P_n is the usual unigram distribution. Therefore P_n is often distorted into P_n^α, with a parameter $0 \leq \alpha \leq 1$. This smoothing helps reducing the range of values of P_n, and α is usually set to 0.75. With this choice, convergence is indeed faster, but the final perplexity is not that better[10]. However,

[8] We also used the same hyperparameter settings than in (Melamud et al., 2017)

[9] Besides the computation time constraints, the batch size is rapidly limited by the GPU memory, while a smaller batch size also increases the computation time.

[10] We should note that for both these experiments, convergence being faster would imply reducing the number of epochs without backtracking, which we didn't do here.

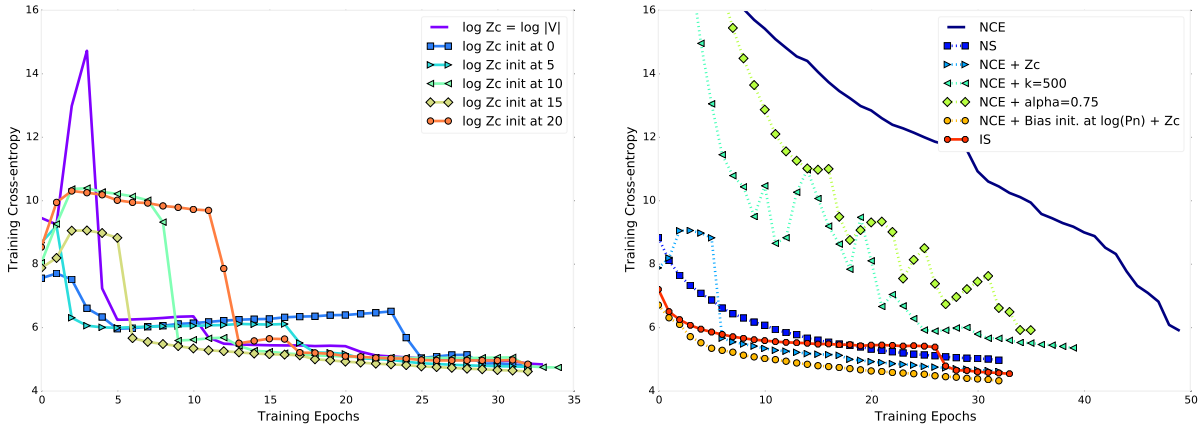

Figure 5: *Left*: Training cross-entropy curves on PTB for un-normalized models trained with NCE while fixing, then learning a parameter Z_c that shifts the partition function, depending on the initial value of Z_c, as presented in section 4.2. *Right*: Training cross-entropy curves on PTB for un-normalized models trained with Importance Sampling, NCE and alternative additions presented in section 4

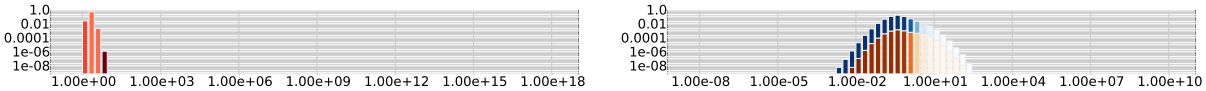

Figure 6: Repartition of the values of the partition function $Z_\theta(H)$ (*left*) and ratio $r_\theta(w|H)$ (*right*) for all training examples (H, w) at initialization, for a model whose output bias is initialized with $\log P_n$.

on the large corpus, this solution fails to stabilize training.

4.2 Shifting the partition function and learning to scale

Following a method introduced by (Chen et al., 2015), we could simply shift all values of the partition function by a fixed value Z_c. In practice, it means we set

$$p_\theta(w|H) = \frac{e^{s_\theta(w,H)}}{Z_c}.$$

Another similar workaround consists in initializing the output bias to $-\log|\mathcal{V}|$ (Vaswani et al., 2013; Melamud et al., 2017). It has the same effect as fixing $Z_c = |\mathcal{V}|$ and initializing the bias to 0. In (Chen et al., 2015) this value is set to $\log Z_c = 9$. In this paper we propose to learn the best shifting value by adding Z_c to the model parameters θ. In this case Z_c can be seen as an extra bias term that learns a normalizing constant. The left graph of figure 5 shows training cross-entropy curves depending on the initial value of Z_c, revealing its impact. On both corpora, learning to scale allows the model to achieve a better perplexity than fixing it to an arbitrary *e.g.* $|\mathcal{V}|$.

4.3 The case of Negative sampling

The *Negative Sampling* (NS) algorithm was popularized by the skip-gram embedding algorithm (Mikolov et al., 2013), and while closely linked to NCE, is not able to directly optimize the likelihood of a language model and learn conditional probabilities, as detailed in (Dyer, 2014). Recently, (Melamud et al., 2016; Melamud et al., 2017) showed Negative Sampling to be viable for language modelling. As previously, we use k samples $(\hat{w}_i)_{1\leq i\leq k}$ from the unigram distribution. The objective maximizes the likelihood of a logistic regression that discriminates true examples from noise samples. However, here, the model directly parametrizes the log-odds, since $P(C = 1|w, H) = \sigma\left(s_\theta^{NS}(w, H)\right)$, which gives the scoring function:

$$J_\theta^H(w) = \log\sigma(s_\theta^{NS}(w, H)) + \sum_{i=1}^{k} \log\sigma(-s_\theta^{NS}(\hat{w}_i, H)) \tag{12}$$

Training method	PTB	1BW Benchmark
MLE	150.2	-
Normalized NCE	159.3	-
NCE	306.0	X
NCE + $\alpha = 0.75$	277.0	X
NCE + $k = 500$	231.9	X
Importance Sampling	168.3	80.2
NS	228.3	99.0
NS + $\alpha = 0.75$	195.8	96.1
NCE + $Z_c = \|\mathcal{V}\|$	178.6	79.2
NCE + $Z_c \in \theta$	172.3	76.6
NCE + Bias init. at $\log(P_n)$	151.8	76.0
NCE + Bias init. at $\log(P_n)$ + $Z_c \in \theta$	148.4	74.7

Table 1: Best testing perplexities obtained on PTB with a full vocabulary, and on the 1 Billion Word Benchmark with a $64K$ words vocabulary. '-' indicates that the models were not trained due to technical limitations, and 'X' indicates that the model did not converge to a perplexity value under $|\mathcal{V}|$.

Therefore, this objective function corresponds to the log-ratio $\log r_\theta(w|H)$ for the NCE. Indeed, we can get back an estimation of the conditional probability using the model score and the noise distribution:

$$\hat{P}(w|H) \propto P_n(w) \exp\left(s_\theta^{NS}(w|H)\right) \tag{13}$$

The model directly learns what is, in the NCE, the ratio of the data and noise distribution, without knowing the values of P_n. The initial values of $r_\theta(w|H) = e^{s_\theta^{NS}(w,H)}$ simply depend on the model initialization, and the initial values of the partition function are close to 1. While the final perplexity values are not among the best[11], we can observe on the bottom graph of figure 5 a far smoother learning curve than that the ones for methods previously presented. More precisely, we avoid the initial increase in cross-entropy that is visible when we fix or learn Z_c (on the top graph). We believe this is due to having an initial model distribution very close to P_n, which is a consequence of multiplying final scores by P_n, as showed in equation 13. To verify this, we trained a NCE model for which we initialized the output bias with values of $\log P_n(w)$. It gave great results, especially on the smaller corpus. By looking at the repartitions of the values of the ratio and partition function at initialization, showed in figure 6, we can check that having $p_\theta(w|H) \approx P_n(w)$ mitigates the main issues described in section 3: first, our distribution is far closer to be normalized; then, the values of $r_\theta(w|H)$ are bundled together around $\frac{1}{k+1}$. This method clearly helps learning; furthermore, when used in complement to learning Z_c, it gives the best model on the PTB. It is however less impactful on the larger corpus. We believe it is due to the fact that the vocabulary is cut at $64K$ words and is thus very small for a corpus of this size: the frequency distribution does not have the Zipfian long tail of rare words, which mitigates the issue we try to solve with this initialization strategy.

4.4 Summary of training recommendations

To have a clearer idea of which solution to use depending on the size of the corpus and of the vocabulary, we experiment with the solutions presented earlier on smaller parts of the 1 Billion Word Benchmark, and with different vocabulary sizes. Testing cross-entropy curves are shown in figure 7. The top figure shows the experiments whose results we presented in table 1, with the full corpus and with $|\mathcal{V}| = 64K$. The two leftmost figures show experiments with 10 and 100 times less data, while the center and rightmost figures show experiments with smaller and larger vocabularies on the full corpus. The main noticeable result is that learning Z_c as a parameter is not working well when we have less data - in this case, initializing the output bias to $-\log|\mathcal{V}|$ is far more efficient, while initializing output bias with values of $\log P_n(w)$ gives even better results. However, this last method loses efficiency when the vocabulary becomes very

[11] Given the recommendation given in (Mikolov et al., 2013) on using $\alpha = 0.75$ with Negative Sampling, we suppose smoothing the noise distribution has an important impact, and added corresponding results, which are indeed better, to table 1.

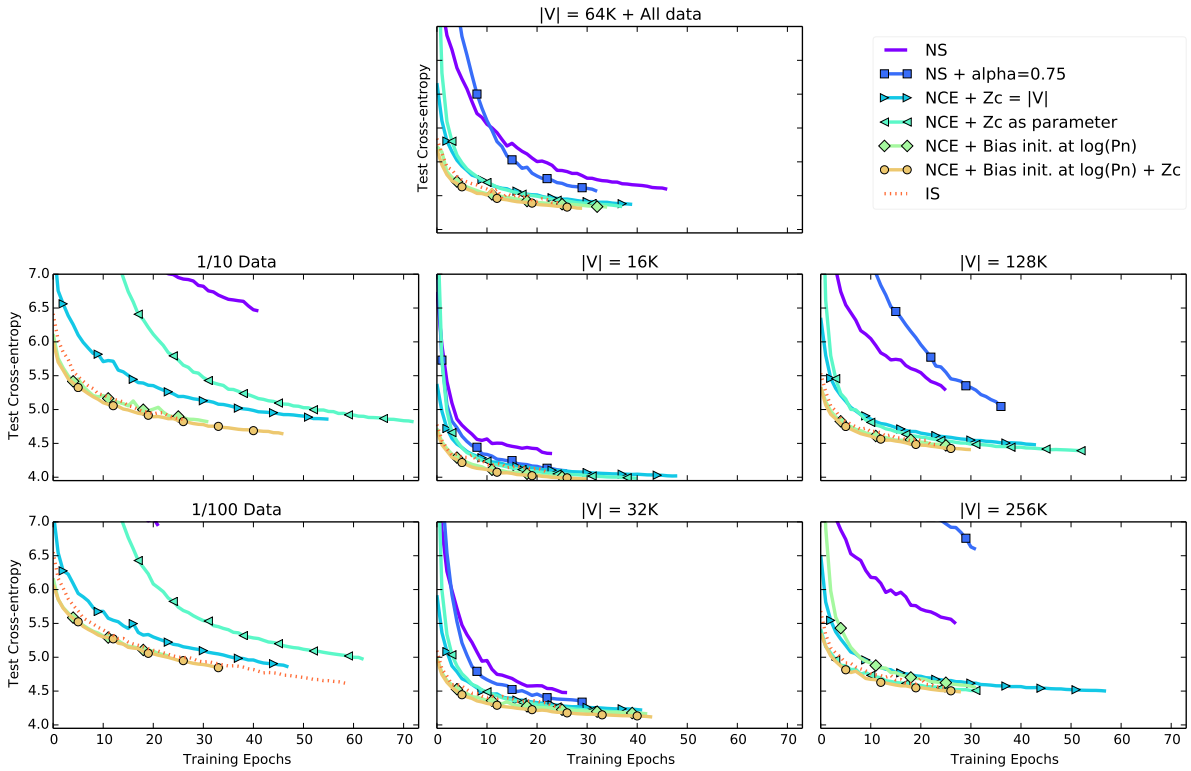

Figure 7: Testing cross-entropy curves on 1BW for un-normalized models trained with Importance Sampling, and NCE-based alternative additions presented in section 4 The seven figures show experiment with various vocabulary sizes $|\mathcal{V}|$ and quantities of Data.

large (here, for $|\mathcal{V}| = 256K$. Finally, combining this initialization scheme and the learning of Z_c as a parameter gives the best results accross all configurations.

5 Conclusion

Training neural language models with large output vocabularies is still challenging. Noise contrastive estimation is one of the most promising training criterion but was empirically shown to be unstable. In this paper, we carried out an extensive analysis of the training process of NCE. We showed how setting the scaling parameters to 1 yields an implicit self-normalization step which necessarily precedes actual learning from data examples. The difficulty of this process depends on the value range of the noise distribution P_n. Given the Zipfian distribution of word frequencies texts, learning with an unigram distribution is getting harder as the vocabulary size grows. This is quite inconsistent with the motivation of this learning criterion. However, we also explored several remedies and showed that when including the scaling parameter in the parameters of the model, the scaling process can be both stable and efficient. Additional improvements can be obtained by initializing the bias of the output layer according to the noise distribution. Combining this 'learning to scale' approach with the adapted initialization scheme yields a very competitive, yet simple, training strategy for neural language models with large vocabularies.

Acknowledgements

We wish to thank the anonymous reviewers for their helpful comments. This work has been funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 645452 (QT21).

References

[Andreas et al.2015] Jacob Andreas, Maxim Rabinovich, Michael I. Jordan, and Dan Klein. 2015. On the accuracy of self-normalized log-linear models. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1783–1791.

[Baltescu and Blunsom2015] Paul Baltescu and Phil Blunsom. 2015. Pragmatic neural language modelling in machine translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 820–829, Denver, Colorado, May–June. Association for Computational Linguistics.

[Bengio and Sénécal2003] Yoshua Bengio and Jean-Sébastien Sénécal. 2003. Quick training of probabilistic neural nets by importance sampling. In *Proceedings of the conference on Artificial Intelligence and Statistics (AISTATS)*.

[Bengio and Sénécal2008] Yoshua Bengio and Jean-Sébastien Sénécal. 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Trans. Neural Networks*, 19(4):713–722.

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

[Chelba et al.2014] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pages 2635–2639.

[Chen et al.2015] Xie Chen, Xunying Liu, Mark J. F. Gales, and Philip C. Woodland. 2015. Recurrent neural network language model training with noise contrastive estimation for speech recognition. In *ICASSP*, pages 5411–5415. IEEE.

[Chen et al.2016] Wenlin Chen, David Grangier, and Michael Auli. 2016. Strategies for training large vocabulary neural language models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1975–1985, Berlin, Germany, August. Association for Computational Linguistics.

[Devlin et al.2014] Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380, Baltimore, Maryland, June. Association for Computational Linguistics.

[Dyer2014] Chris Dyer. 2014. Notes on noise contrastive estimation and negative sampling. *CoRR*, abs/1410.8251.

[Gutmann and Hirayama2011] Michael Gutmann and Junichiro Hirayama. 2011. Bregman divergence as general framework to estimate unnormalized statistical models. In *UAI*.

[Gutmann and Hyvärinen2010] Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, pages 297–304.

[Gutmann and Hyvärinen2013] M.U. Gutmann and A. Hyvärinen. 2013. Estimation of unnormalized statistical models without numerical integration. In *Proceedings of the Workshop on Information Theoretic Methods in Science and Engineering*.

[Jean et al.2015] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China, July. Association for Computational Linguistics.

[Józefowicz et al.2016] Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

[Le et al.2011] Hai-Son Le, Ilya Oparin, Alexandre Allauzen, Jean-Luc Gauvain, and Francois Yvon. 2011. Structured output layer neural network language model. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP)*, pages 5524–5527, Prague, Czech Republic.

[Melamud et al.2016] Oren Melamud, Ido Dagan, and Jacob Goldberger. 2016. PMI matrix approximations with applications to neural language modeling. *CoRR*, abs/1609.01235.

[Melamud et al.2017] Oren Melamud, Ido Dagan, and Jacob Goldberger. 2017. A simple language model based on PMI matrix approximations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1861–1866, Copenhagen, Denmark, September. Association for Computational Linguistics.

[Mikolov et al.2010] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

[Mikolov et al.2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

[Mnih and Hinton2009] Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1081–1088. Curran Associates, Inc.

[Mnih and Teh2012] Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*.

[Morin and Bengio2005] Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 246–252. Society for Artificial Intelligence and Statistics.

[Pihlaja et al.2012] Miika Pihlaja, Michael Gutmann, and Aapo Hyvärinen. 2012. A family of computationally efficient and simple estimators for unnormalized statistical models. *CoRR*, abs/1203.3506.

[Schwenk2007] Holger Schwenk. 2007. Continuous space language models. *Comput. Speech Lang.*, 21(3):492–518, July.

[Vaswani et al.2013] Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1387–1392, Seattle, Washington, USA, October. Association for Computational Linguistics.

[Zoph et al.2016] Barret Zoph, Ashish Vaswani, Jonathan May, and Kevin Knight. 2016. Simple, fast noise-contrastive estimation for large RNN vocabularies. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1217–1222, San Diego, California, June. Association for Computational Linguistics.