A MATRIX APPROXIMATION VIEW OF NCE THAT JUSTIFIES SELF-NORMALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-normalizing discriminative models approximate the normalized probability of a class without having to compute the partition function. This property is useful to computationally-intensive neural network classifiers, as the cost of computing the partition function grows linearly with the number of classes and may become prohibitive. In particular, since neural language models may deal with up to millions of classes, their self-normalization properties received notable attention. Several recent studies *empirically* found that language models, trained using Noise Contrastive Estimation (NCE), exhibit self-normalization, but could not explain why. In this study, we provide a theoretical justification to this property by viewing NCE as a low-rank matrix approximation. Our empirical investigation compares NCE to the alternative explicit approach for self-normalizing language models. It also uncovers a surprising negative correlation between self-normalization and perplexity, as well as some regularity in the observed errors that may potentially be used for improving self-normalization algorithms in the future.

1 INTRODUCTION

The ability of statistical language models (LMs) to estimate the probability of a word given a context of preceding words, plays an important role in many NLP tasks, such as speech recognition and machine translation. Recurrent Neural Network (RNN) language models have recently become the preferred method of choice, having outperformed traditional n-gram LMs across a range of tasks (Jozefowicz et al. (2016)). Unfortunately however, they suffer from scalability issues incurred by the computation of the softmax normalization term, which is required to guarantee proper probability predictions. The cost of this computation is linearly proportional to the size of the word vocabulary and has a significant impact on both training and testing.[1]

Several methods have been proposed to cope with this scaling issue by replacing the softmax with a more computationally efficient component at train time. These include importance sampling (Bengio & et al (2003)), hierarchical softmax (Minh & Hinton (2008), BlackOut (Ji et al. (2016)) and Noise Contrastive Estimation (NCE) (Gutmann & Hyvarinen (2012)). NCE has been applied to train neural LMs with large vocabularies (Mnih & Teh (2012)) and more recently was also successfully used to train LSTM-RNN LMs (Vaswani et al. (2013); Chen et al. (2015); Zoph et al. (2016)), achieving near state-of-the-art performance on language modeling tasks (Jozefowicz et al. (2016); Chen et al. (2016)). All the above works focused on solving the run-time complexity problem at train time. However, at test time the assumption was that one still needs to explicitly compute the softmax normalization term to obtain a normalized score fit as an estimate for the probability of a word.

Self-normalization was recently proposed as means to address the high run-time complexity associated with predicting normalized probabilities at test time. A self-normalized discriminative model is trained to produce near-normalized scores in the sense that the sum over the scores of all classes is approximately one. If this approximation is close enough, the assumption is that the costly exact normalization can be waived at test time without significantly sacrificing prediction accuracy (Devlin et al. (2014)). Two main approaches were proposed to train self-normalizing models. *Explicit self-normalization* is based on using softmax for training and explicitly encouraging the normalization term of the softmax to be as close to one as possible, thus making its computation redundant at test

[1] Alleviating this problem using sub-word representations is a parallel line of research not discussed here.

time (Devlin et al. (2014); Andreas & Klein (2015); Chen et al. (2016)). The alternative approach is based on NCE. The original formulation of NCE included a normalization term Z. However, the first work that applied NCE to LM (Mnih & Teh (2012)) discovered, *empirically*, that fixing Z to a constant did not affect the performance. More recent studies (Vaswani et al. (2013); Zoph et al. (2016); Chen et al. (2015); Oualil & Klakow (2017)) *empirically* found that models trained using NCE with a fixed Z, exhibit self-normalization, but they could not explain this behavior. To the best of our knowledge, the only theoretical analysis of self-normalization was proposed by Andreas & Klein (2015). This analysis shows that a model trained explicitly to be self-normalizing only on a subset of the training instances, can potentially be self-normalizing on other similar instances as well. However, their analysis cannot explain how NCE can be self-normalizing without explicitly imposing self-normalization on any of its training instances.

The main contribution of this study is providing a theoretical justification to the self-normalization property of NCE, which was empirically observed in prior work. We do so by showing that NCE's unnormalized objective can be viewed as finding the best low-rank approximation of the normalized conditional probabilities matrix, without having to explicitly estimate the partition function. While the said self-normalizing property of NCE is more general, we focus the empirical contribution of the paper on language modeling. We investigate the self-normalization performance of NCE as well as that of the alternative explicit self-normalization approach over two datasets. Our results suggest, somewhat surprisingly, that models that achieve better perplexities tend to have worse self-normalization properties. We also observe that given a context, the sum of the self-normalized scores is negatively correlated with the entropy of the respective normalized distribution.

2 Viewing NCE as a Matrix Factorization

In this section, we first review the NCE algorithm for language modeling and then introduce an interpretation of it as a matrix factorization procedure. Noise Contrastive Estimation (NCE) is a popular algorithm for efficiently training language models. NCE transforms the parameter learning problem into a binary classifier training problem. Let $p(w|c)$ be the probability of a word w given a context c that represents its entire preceding context, and let $p(w)$ be a 'noise' word distribution (e.g. a unigram distribution). The NCE approach assumes that the word w is sampled from a mixture distribution $\frac{1}{k+1}(p(w|c) + kp(w))$ such that the noise samples are k times more frequent than samples from the 'true' distribution $p(w|c)$. Let y be a binary random variable such that $y = 0$ and $y = 1$ correspond to a noise sample and a true sample, respectively, i.e. $p(w|c, y = 0) = p(w)$ and $p(w|c, y = 1) = p(w|c)$. Assume the distribution $p(w|c)$ has the following parametric form:

$$p_{nce}(w|c) = \frac{1}{Z_c} \exp(\vec{w} \cdot \vec{c} + b_w) \tag{1}$$

such that $\vec{w}$ and $\vec{c}$ are d-dimensional vector representations of the word w and its context c and Z_c is a normalization term. Applying Bayes' rule, it can be easily verified that:

$$p_{nce}(y = 1|w, c) = \sigma(\vec{w} \cdot \vec{c} + b_w - \log Z_c - \log(p(w)k)) \tag{2}$$

where $\sigma()$ is the sigmoid function.

NCE uses Eq. (2) and the following objective function to train a binary classifier that decides which distribution was used to sample w:

$$S_{nce} = \sum_{w,c \in D} \Big[\log p(1|w, c) + \sum_{i=1}^{k} \log p(0|u_i, c) \Big] \tag{3}$$

such that w, c go over all the word-context co-occurrences in the learning corpus D and $u_1, ..., u_k$ are 'noise' samples drawn from the word unigram distribution.

The normalization factor Z_c is not a free parameter and to obtain its value, one needs to compute $Z_c = \sum_{v \in V} \exp(\vec{v} \cdot \vec{c} + b_v)$ for each context c, where V is the word vocabulary. The original NCE paper (Gutmann & Hyvarinen (2012)) proposed to learn the normalization term during training and then use it to normalize the model at test time. In language modeling applications, this computation

is typically not feasible due to the exponentially large number of possible contexts. Computing the value of Z_c at test time is possible, though expensive due to the large vocabulary size. Mnih & Teh (2012) found empirically that setting $Z_c = 1$ at train time, which removes the explicit normalization constraint in the NCE formulation (1), didn't affect the performance of the resulting model. At test time, to compute $\log p(w|c)$, they still had to normalize the score $\vec{w} \cdot \vec{c} + b_w$, by explicitly computing Z_c over all the vocabulary words, in order to obtain a proper distribution.

We next present an alternative interpretation of the NCE language modeling algorithm as a low-rank matrix approximation. This view of NCE makes the normalization factor redundant during training and explains the self-normalization property, as was empirically observed in later works (Vaswani et al. (2013); Zoph et al. (2016); Chen et al. (2015); Oualil & Klakow (2017)).

Definition: The Pointwise Conditional Entropy (PCE) matrix of a conditional word distribution $p(w|c)$ is:

$$\text{pce}(w, c) = \log p(w|c)$$

where w goes over the words in the vocabulary and c goes over all the left (preceding) side contexts.

The NCE modeling (1) can also be written as a matrix $m(w, c) = \log p_{nce}(w|c) = \vec{w} \cdot \vec{c} + b_w - \log Z_c$ with the same dimensions as the PCE matrix. Assuming that $\vec{w}$ and $\vec{c}$ are d-dimensional vectors, the rank of the matrix m is at most $d + 2$.

Let $p(w, c)$ be the joint distribution of words and their left side contexts. The NCE score (3) can be viewed as a corpus-based approximation of the following expectation based score:

$$S_{nce}(m) = \sum_{w,c} p(w, c) \log \sigma\Big(m(w, c) - \log(p(w)k)\Big) \tag{4}$$

$$+k \sum_{w,c} p(w)p(c) \log \Big(1 - \sigma\Big(m(w, c) - \log(p(w)k)\Big)\Big).$$

When we actually compute the NCE score based on a given corpus, we replace the expectation in the first term by averaging over the corpus and the expectation in the second term is replaced by sampling of negative examples from the word unigram distribution.

Theorem 1: The NCE score $S_{nce}(m)$ (4) obtains its global maximum at the PCE matrix. In other words, $S_{nce}(m) \leq S_{nce}(\text{pce})$ for every matrix m.

Proof: The claim can be easily verified by computing the derivative and set it to zero. An alternative proof is based on the fact that the word2vec NEG cost function obtains its global maximum at the Pointwise Mutual Information (PMI) matrix. (Levy & Goldberg (2014)) showed that the function

$$f(x) = p(w, c) \log \sigma(x) + kp(w)p(c) \log(1 - \sigma(x))$$

obtains its global maximum when $x = \text{pmi}_k(w, c) = \log \frac{p(w,c)}{kp(w)p(c)}$. In our case it implies that the global maximum of (4) is obtained when $m(w, c) - \log(p(w)k) = \text{pmi}_k(w, c)$. Observing that $\text{pmi}(w, c) = \text{pce}(w, c) - \log(p(w)k)$ we complete the proof. □

We next show that the value of the function $S_{nce}(m)$ at its maximum point, the PCE matrix, has a concrete interpretation. The Kullback-Leibler (KL) divergence of a distribution p from a distribution q is defined as follows: $KL(p||q) = \sum_{i \in \mathcal{A}} p_i \log \frac{p_i}{q_i}$. The Jensen-Shannon (JS) divergence (Lin (1991)) between distributions p and q is:

$$\text{JS}_\alpha(p, q) = \alpha \text{KL}(p||r) + (1-\alpha)\text{KL}(q||r) \tag{5}$$

such that $0 < \alpha < 1$, $r = \alpha p + (1 - \alpha)q$. Unlike KL divergence, JS divergence is bounded from above and $0 \leq \text{JS}_\alpha(p, q) \leq 1$. It can be easily verified that the value of the NCE score with k negative samples (4) at the PCE matrix satisfies:

$$S_{nce}(\text{pce}) = (k+1) \cdot JS_\alpha(p(w, c), p(c)p(w))$$

where $\alpha = \frac{1}{k+1}$. In other words the global optimum of the NCE score is the Jensen-Shannon divergence between the joint distribution of words and their left-side contexts and the product of their marginal distributions. The NCE algorithm finds the best d-dimensional approximation m of the PCE matrix in the sense that it minimizes the difference $(k+1) \cdot JS_\alpha(p(w, c), p(c)p(w)) - S_{nce}(m)$.

In the traditional derivation of NCE, the parametric model is used to define a proper conditional probability: $p_{nce}(w|c) = \frac{1}{Z_c}\exp(\vec{w}\cdot\vec{c}+b_w)$. Hence, to guarantee that we need a normalization factor Z_c for each context c. In our NCE interpretation, the training goal is to find the best *unnormalized* low-rank approximation of the PCE matrix. Hence, no normalization factor is involved. Although, normalization is not explicitly included in our view of NCE, we have shown that even so, our model attempts to approximate the true conditional probabilities, which are normalized, and hence we can provide guarantees as to its self-normalization properties as we describe in the next section.

Finally, revisiting prior work, Mnih & Teh (2012); Vaswani et al. (2013); Zoph et al. (2016) used $Z = 1$, while Chen et al. (2015); Oualil & Klakow (2017) reported that using $\log(Z) = 9$ at train time gave them the best results with their NCE implementation. Setting a specific fixed value for Z would alter the mean input to the sigmoid function in the NCE score, which may ensure that it is closer to zero. This can potentially improve training stability, convergence speed and performance in a way similar to batch normalization (Ioffe & Szegedy (2015)). However, we note that it is not related to distribution normalization.

3 THE NCE SELF-NORMALIZATION PROPERTY

At test time, when we use the model learned by NCE to compute the conditional probability $p_{nce}(w|c)$ (1), we need to compute the normalization factor:

$$Z_c = \sum_w \exp(m(w,c)) = \sum_w \exp(\vec{w}\cdot\vec{c}+b_w). \tag{6}$$

Note that the NCE language model obtained from the PCE matrix by setting $m(w,c) = \text{pce}(w,c) = \log(w|c)$, is clearly self normalized:

$$Z_c = \sum_w \exp(\text{pce}(w,c)) = \sum_w p(w|c) = 1.$$

Theorem 1 showed that the NCE algorithm finds the best low-rank unnornmalized matrix approximation of the PCE matrix. Hence, we can assume that the matrix m is close to the PCE matrix and therefore the normalization factors of the LM based on m should be also close to 1:

$$\sum_w \exp(m(w,c)) \approx \sum_w \exp(\text{pce}(w,c)) = 1. \tag{7}$$

We next formally show that if the matrix m is close to the PCE matrix then the NCE model defined by m is approximately self-normalized.

Theorem 2: Assume that for a given context c there is an $0 < \epsilon$ such that $|m(w,c) - \text{pce}(w,c)| \leq \epsilon$ for every $w \in V$. Then $|\log Z_c| \leq \epsilon$.

Proof:

$$\log Z_c = \log \sum_w \exp(\vec{w}\cdot\vec{c}+b_w - \log p(w|c) + \log p(w|c))$$
$$= \log \sum_w (p(w|c)\exp(\vec{w}\cdot\vec{c}+b_w - \log p(w|c))).$$

Given the assumption that $|m(w,c) - \text{pce}(w,c)| = |\vec{w}\cdot\vec{c}+b_w - \log p(w|c)| \leq \epsilon$, we obtain that:

$$\log Z_c \leq \log \sum_w (p(w|c)\exp(\epsilon)) = \epsilon. \tag{8}$$

The concavity of the log function implies that:

$$\log Z_c \geq \sum_w p(w|c)\log\exp(\vec{w}\cdot\vec{c}+b_w - \log p(w|c)) \geq \sum_w p(w|c)(-\epsilon) = -\epsilon. \tag{9}$$

Combining Eq. (8) and Eq. (9) we finally obtain that $|\log Z_c| \leq \epsilon$. □

In the appendix we show that Theorem 2 remains true if we replace the assumption $\max_{w\in V}|m(w,c) - \text{pce}(w,c)| \leq \epsilon$ by the weaker and more realistic assumption that $\log\sum_{w\in V} p(w|c)\exp(|m(w,c) - \text{pce}(w,c)|) \leq \epsilon$.

To summarize, in our analysis we first show (Theorem 1) that the NCE training goal is to make the *unnormalized* score $(\vec{w} \cdot \vec{c} + b_w)$ close to the normalized $\log p(w|c)$. We then show (Theorem 2) that if the unnormalized score is indeed close to $\log p(w|c)$, then $\log Z_c$ is close to zero. Combining the two theorems, we obtain that the LM learned by NCE is self-normalized.

4 Relations to Other Language Models

In this section we address two related language models and briefly describe how our analysis is related to their training strategies. The standard LM learning method, which is based on a softmax output layer, is not self-normalized. Devlin et al. (2014) proposed to explicitly encourage self-normalization as part of its training objective function by penalizing deviation from self-normalizing:

$$S_{Dev} = \sum_{w,c \in D} \Big[(\vec{w} \cdot \vec{c} + b_w - \log Z_c) - \alpha (\log Z_c)^2 \Big] \tag{10}$$

where $Z_c = \sum_{v \in V} \exp(\vec{v} \cdot \vec{c} + b_v)$ and α is a constant. The drawback of this approach is that at least at train time you need to explicitly compute the costly Z_c. Andreas & Klein (2015) proposed an efficiently computed approximation of (10) by eliminating Z_c in the first term and computing the second term only on a sampled subset D' of the corpus D:

$$S_{And} = \sum_{w,c \in D} (\vec{w} \cdot \vec{c} + b_w) - \frac{\alpha}{\gamma} \sum_{c \in D'} (\log Z_c)^2 \tag{11}$$

where γ is an additional constant that determines the sampling rate. They also provided analysis that justifies computing Z_c only on a subset of the corpus by showing that if a given LM is exactly self-normalized on a dense set of contexts (i.e. each context c is close to a context c' s.t. $\log Z_{c'} = 0$) then $E|\log Z_c|$ is small. This could justifies computing Z_c only on a small subset of contexts, but cannot explain why NCE training produces a self-normalized model without computing Z_c on any context at all.

Importance sampling (IS) (Bengio & et al (2003)) is an efficient alternative to a full softmax layer that is closely related to NCE. In IS we assume that for each context c we are given $k+1$ words $w = u_0, u_1, ..., u_k$ that were sampled in the following way. First we sample a uniform random variable $y \in \{0, ..., k\}$. Then for $y = i$ we sample u_i according to $p(\cdot|c)$ and all the other k words are sampled from the unigram distribution:

$$p(u_0, ..., u_k | c, y = i) = p(u_i|c) \prod_{j \neq i} p(u_j) = \frac{p(u_i|c)}{p(u_i)} \prod_{j=0}^{k} p(u_j).$$

Given the observation that for all the contexts $y = 0$, (i.e. $u_1, ..., u_k$ are sampled from the word unigram distribution and $u_0 = w$ is sampled from $p(w|c)$), the IS objective function is:

$$S_{\text{is}} = \sum_{w,c \in D} \Big[(\vec{w} \cdot \vec{c} + b_w - \log Z_c) - \log p(w) - \log(\sum_{i=0}^{k} \frac{\exp(\vec{u}_i \cdot \vec{c} + b(u_i) - \log Z_c)}{p(u_i)}) \Big]. \tag{12}$$

It can be easily verified that the normalization factor Z_c is canceled out in the IS objective function. Hence, there is no need to estimate it in training. Unlike NCE, the network learned by IS was not found to be self-normalized. As a result, explicit computation of the normalization factor is required at test time (Oualil & Klakow (2017)).

5 Experiments

In this section, we empirically investigate the self-normalization properties of NCE language modeling and its explicit self-normalization alternative.

5.1 Experimental Settings

We investigated LSTM-based language models with two alternative training schemes: (1) ***NCE-LM*** - the NCE language model described in Section 2; and (2) ***DEV-LM*** - the softmax language model

	NCE-LM			*SM-LM*		
d	μ_z	σ_z	perp	μ_z	σ_z	perp
	PTB validation set					
30	-0.18	0.11	267.6	2.29	0.97	243.4
100	-0.19	0.17	150.9	3.03	1.52	145.2
300	-0.15	0.29	100.1	3.77	1.98	97.7
650	-0.17	0.37	87.4	4.36	2.31	87.3
	WIKI validation set					
30	-0.20	0.13	357.4	2.57	1.02	322.2
100	-0.24	0.19	194.3	3.34	1.45	191.1
300	-0.23	0.27	125.6	4.19	1.73	123.3
650	-0.23	0.35	110.5	4.67	1.83	110.7

Table 1: Self-normalization and perplexity results of the self-normalizing NCE-LM against the non-self-normalizing standard softmax LM (SM-LM). d denotes the size of the compared models (units).

with explicit self-normalization proposed by Devlin et al. (2014). Following Devlin et al. (2014), to make both DEV-LM and NCE-LM approximately self-normalized at init time, we initialized their output bias terms to $b_w = -\log|V|$, where V is the word vocabulary. We set the negative sampling parameter for NCE-LM to $k = 100$, following Zoph et al. (2016), who showed highly competitive performance with NCE LMs trained with this number of samples, and Melamud et al. (2017). Except where noted otherwise, other details of our implementation and choice of hyperparameters follow Zaremba et al. (2014) who achieved strong perplexity results using standard LSTM-based neural language models. Specifically, we used a 2-layer LSTM with a 50% dropout ratio to represent the preceding (left-side) context of a predicted word.[2] All models were implemented using the Chainer toolkit (Tokui et al. (2015)).

We used two language modeling datasets in the evaluation. The first dataset, denoted *PTB*, is a version of the Penn Tree Bank, commonly used to evaluate language models.[3] It consists of 929K/73K/82K training/validation/test words respectively and has a 10K word vocabulary. The second dataset, denoted *WIKI*, is the WikiText-2, more recently introduced by Merity et al. (2016). This dataset was extracted from Wikipedia articles and is somewhat larger, with 2,088K/217K/245K train/validation/test tokens, respectively, and a vocabulary size of 33K.

To evaluate self-normalization, we look at two metrics: (1) $\mu_z = E(\log(Z_c))$, which is the mean log value of the normalization term, across contexts c in a dataset C; and (2) $\sigma_z = \sigma(\log(Z_c))$, which is the corresponding standard deviation. The closer these two metrics are to zero, the more self-normalizing the model is considered to be. We note that a model with $|\mu_z| >> 0$ can potentially be 'corrected' (as we show later) by subtracting μ_z from the unnormalized score. However, this is not the case for σ_z. Therefore, from a practical point of view, we consider σ_z to be the more important metric of the two. In addition, we also look at the classic perplexity metric, which is considered a standard measure for the quality of the model predictions. Importantly, when measuring perplexity, except where noted otherwise, we first perform exact normalization of the models' unnormalized scores by computing the normalization term.

5.2 Results

Table 1 shows a range of results that we got on the validation sets when evaluating NCE-LM against standard softmax language model baseline (*SM-LM*).[4] Looking at the results, we can first see that consistently with previous works, NCE-LM is approximately self-normalized as apparent by relatively low $|\mu_z|$ and σ_z values. On the other hand, SM-LM, as expected, is far from being self-normalized. In terms of perplexity, we see that SM-LM performs a little better when model dimensionality is low, but the gap closes entirely at $d = 650$. Curiously, while perplexity improves with

[2]For more details, we refer the reader to the description of the medium LSTM model in Zaremba et al. (2014). We used only 20 training iterations instead of 39, since all models seemed to have converged by then.

[3]Available from Tomas Mikolov at: `http://www.fit.vutbr.cz/~imikolov/rnnlm/simple-examples.tgz`

[4]More specifically, SM-LM is exactly DEV-LM with $\alpha = 0$.

	DEV-LM								
	$\alpha = 0.1$			$\alpha = 1.0$			$\alpha = 10.0$		
d	μ_z	σ_z	perp	μ_Z	σ_z	perp	μ_z	σ_z	perp
	PTB validation set								
30	-0.12	0.21	242.6	-0.16	0.09	250.9	-0.13	0.060	307.2
100	-0.10	0.28	143.3	-0.17	0.11	149.5	-0.12	0.058	182.0
300	-0.09	0.36	96.3	-0.14	0.14	100.8	-0.16	0.054	121.3
650	-0.14	0.43	85.0	-0.17	0.18	86.3	-0.11	0.071	99.5
	WIKI validation set								
30	-0.10	0.23	334.1	-0.17	0.08	338.7	-0.15	0.055	389.0
100	-0.13	0.28	189.4	-0.22	0.13	191.1	-0.15	0.071	228.3
300	-0.15	0.34	121.9	-0.20	0.17	125.7	-0.13	0.081	143.6
650	-0.23	0.42	109.1	-0.23	0.20	110.0	-0.12	0.089	116.9

Table 2: Self-normalization and perplexity results of the self-normalizing DEV-LM for different values of the normalization factor α. d denotes the size of the compared models (units).

	NCE-LM				*DEV-LM* ($\alpha = 1.0$)			
dataset	μ_z	σ_z	perp	u-perp	μ_z	σ_z	perp	u-perp
PTB-test	-0.004	0.35	83.7	83.4	-0.001	0.17	83.1	83.0
WIKI-test	0.003	0.36	104.3	104.6	0.002	0.20	104.1	104.2

Table 3: Self-normalization and perplexity results on test sets for 'shifted' models with $d = 650$. 'u-perp' denotes unnormalized perplexity.

higher dimensionality, we see that the quality of NCE-LM's self-normalization, as evident particularly by σ_z, actually degrades. This is surprising, as we would expect that stronger models with more parameters would approximate $p(w|c)$ more closely. We further investigate this behavior in Section 5.3.

Next, Table 2 compares the self-normalization and perplexity performance of DEV-LM for different values of the constant α on the validation sets. As could be expected, the larger the value of α is, the better the self-normalization becomes, reaching very good self-normalization for $\alpha = 10.0$. On the other hand, the improvement in self-normalization seems to occur at the expense of perplexity. This is particularly true for the smaller models,[5] but is still evident even for $d = 650$. Interestingly, as with NCE-LM, we see that σ_z is growing (i.e. self-normalization becomes worse), with the size of the model, and is negatively correlated with the improvement in perplexity.

Finally, for the test-set evaluation, we propose a simple technique to center the $\log(Z)$ values of a self-normalizing model's scores around zero. Let μ_z be $E(\log(Z))$ observed on the validation set at train time. The probability estimates of the 'shifted' model are $\log p(w|c) = \vec{w} \cdot \vec{c} + b_w - \mu_z$. Table 3 shows the results that we get when evaluating the shifted NCE-LM and DEV-LM models with $d = 650$. For DEV-LM, we chose $\alpha = 1.0$, which seems to provide an optimal trade-off between self-normalization and perplexity performance on the validation sets. Following Oualil & Klakow (2017), in addition to perplexity, we also report 'unnormalized perplexity', which is computed with the unnormalized conditional probability estimates. When the perplexity measure is close to the unnormalized perplexity, this suggests that the unnormalized estimates are in fact nearly normalized. As can be seen, with the shifting method, both models achieve near perfect (zero) μ_z value, and their unnormalized perplexities are almost identical to their respective real perplexities. Also, comparing the perplexities of NCE-LM to those of DEV-LM, we see near identical performance. The standard deviation of the normalization term of DEV-LM is notably better than that of NCE-LM. However, we note that NCE's advantage is that unlike DEV-LM, it doesn't include the normalization term in its training objective, and therefore its training time does not grow with the size of the vocabulary.

[5]We also experimented with $\alpha = 0.01$, but this yielded worse self-normalization with no perplexity benefits and therefore these results were omitted for brevity.

	PTB-validation		*WIKI-validation*	
d	NCE-LM	DEV-LM ($\alpha = 1.0$)	NCE-LM	DEV-LM ($\alpha = 1.0$)
30	-0.33	-0.27	-0.50	-0.26
100	-0.29	-0.29	-0.53	-0.49
300	-0.46	-0.41	-0.56	-0.63
650	-0.50	-0.45	-0.53	-0.64

Table 4: Pearson's correlation between entropy and $\log(Z)$ on samples from the validation sets.

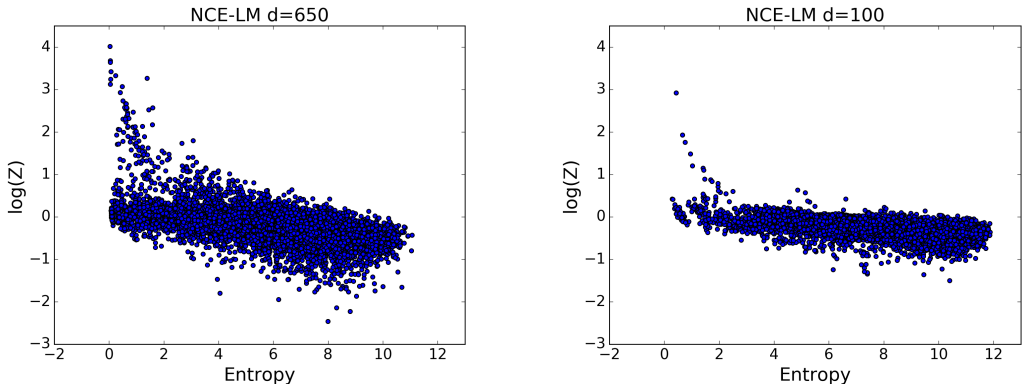

Figure 1: The normalization term of a predicted distribution as a function of its entropy on a sample from the WIKI validation set.

5.3 Analysis

The entropy of the distributions predicted by a language model is a measure of how uncertain it is regarding the identity of the predicted word. Low-entropy distributions would be concentrated around few possible words, while high-entropy ones would be much more spread out. To more carefully analyze the self-normalization properties of NCE-LM, we computed the Pearson's correlation between the entropy of a predicted distribution $H_c = -\sum_v p(v|c) \log p(v|c)$ and its normalization term, $\log(Z_c)$. As can be seen in Table 4, it appears that a regularity exists, where the value of $\log(Z_c)$ is negatively correlated with entropy. Furthermore, it seems that, to an extent, the correlation is stronger for larger models. To illustrate this phenomenon, we plot a sample of the predicted distributions in Figure 1. We can see there in particular, that low entropy distributions can be associated with very high values of $\log(Z_c)$, deviating a lot from the self-normalization objective of $\log(Z_c) = 0$. Examples for contexts for which NCE-LM predicts such distributions are: "During the American Civil [War]" and "The United [States]", where the actual word following the preceding context appears in parenthesis. We hypothesize that this observation could be a contributing factor to our earlier finding that larger models have larger variance in their normalization terms, though it seems to account only for some of that at best. Furthermore, we hope that this regularity could be exploited to improve self-normalization algorithms in the future.

6 Conclusions

We provided theoretical justification to the empirical observation that NCE is self-normalizing. Our empirical investigation shows that it performs reasonably well, but not as good as a language model that is explicitly trained to self-normalize. Accordingly, we believe that an interesting future research direction could be to augment NCE's training objective with some explicit self-normalization component. In addition, we revealed unexpected correlations between self-normalization and perplexity performance, as well as between the partition function of self-normalized predictions and the entropy of the respective distribution. We hope that these insights would be useful in improving self-normalizing models in future work.

REFERENCES

J. Andreas and D. Klein. When and why are log-linear models self-normalizing? In *NAACL*, 2015.

Y. Bengio and J. Senecal et al. Quick training of probabilistic neural nets by importance sampling. In *AISTATS*, 2003.

W. Chen, D. Grangier, and M. Auli. Strategies for training large vocabulary neural language models. *CoRR*, abs/1512.04906, 2016.

X. Chen, X. Liu, M. Gales, and P. C. Woodland. Recurrent neural network language model training with noise contrastive estimation for speech recognition. In *ICASSP*, 2015.

J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. Schwartz, and J. Makhoul. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of ACL*, 2014.

M. U. Gutmann and A. Hyvarinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13:307–361, 2012.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

S. Ji, S. Vishwanathan, N. Satish, A. Nadathur, J. Michael, and P. Dubey. Blackout: Speeding up recurrent neural network language models with very large vocabularies. ICLR, 2016.

R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

O. Levy and Y. Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, 2014.

J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.

O. Melamud, I. Dagan, and J. Goldberger. A simple language model based on pmi matrix approximations. In *EMNLP*, 2017.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

A. Minh and G. E. Hinton. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems*, 2008.

A. Mnih and Y. W. Teh. A fast and simple algorithm for training neural probabilistic language models. In *ICML*, 2012.

Y. Oualil and D. Klakow. A batch noise contrastive estimation approach for training large vocabulary language models. In *Interspeech*, 2017.

S. Tokui, K. Oono, S. Hido, and J. Clayton. Chainer: a next-generation open source framework for deep learning. In *Workshop on Machine Learning Systems (LearningSys) in The 29th Annual Conference on Neural Information Processing Systems*, 2015.

A. Vaswani, Y. Zhao, V. Fossum, and D. Chiang. Decoding with large-scale neural language models improves translation. In *EMNLP*, 2013.

W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

B. Zoph, A. Vaswani, J. May, and K. Knight. Simple, fast noise-contrastive estimation for large RNN vocabularies. In *NAACL*, 2016.

APPENDIX

Theorem A1: Assume that for a given context c there is an $0 < \epsilon$ such that

$$\log \sum_{w \in V} p(w|c) \exp(|m(w,c) - \log p(w|c)|) \leq \epsilon.$$

Let $Z_c = \sum_w \exp(m(w,c))$. Then $|\log Z_c| \leq \epsilon$.

Proof:

$$\log Z_c = \log \sum_w \exp(\vec{w} \cdot \vec{c} + b_w - \log p(w|c) + \log p(w|c))$$

$$= \log \sum_w (p(w|c) \exp(m(w,c) - \log p(w,c)))$$

$$\leq \log \sum_{w \in V} p(w|c) \exp(|m(w,c) - \log p(w,c)|) \leq \epsilon. \tag{13}$$

The concavity of the log function implies that:

$$-\log Z_c \leq -\sum_w p(w|c) \log \exp(\vec{w} \cdot \vec{c} + b_w - \log p(w|c)) \tag{14}$$

$$= \sum_w p(w|c)(-(\vec{w} \cdot \vec{c} + b_w - \log p(w|c)))$$

The concavity of the log function implies that:

$$\leq \log \sum_w p(w|c) \exp(-(\vec{w} \cdot \vec{c} + b_w - \log p(w|c)))$$

$$\leq \log \sum_w p(w|c) \exp(|m(w,c) - \log p(w|c)|) \leq \epsilon$$

Combining Eq. (13) and Eq. (14) we finally obtain that $|\log Z_c| \leq \epsilon$. □

We can also state a global version of Theorem A.1 and its proof is similar.

Theorem A2: Assume that there is an $0 < \epsilon$ such that

$$\log \sum_{w,c} p(w,c) \exp(|m(w,c) - \log p(w|c)|) \leq \epsilon.$$

Then $|\sum_c p(c) \log Z_c| \leq \epsilon$.