Deep Neural Language Models for Machine Translation

Minh-Thang Luong **Michael Kayser** **Christopher D. Manning**
Computer Science Department, Stanford University, Stanford, CA, 94305
{lmthang, mkayser, manning}@stanford.edu

Abstract

Neural language models (NLMs) have been able to improve machine translation (MT) thanks to their ability to generalize well to long contexts. Despite recent successes of deep neural networks in speech and vision, the general practice in MT is to incorporate NLMs with only one or two hidden layers and there have not been clear results on whether having more layers helps. In this paper, we demonstrate that deep NLMs with three or four layers outperform those with fewer layers in terms of both the perplexity and the translation quality. We combine various techniques to successfully train deep NLMs that jointly condition on both the source and target contexts. When reranking n-best lists of a strong web-forum baseline, our deep models yield an average boost of 0.5 TER / 0.5 BLEU points compared to using a shallow NLM. Additionally, we adapt our models to a new sms-chat domain and obtain a similar gain of 1.0 TER / 0.5 BLEU points.[1]

1 Introduction

Deep neural networks (DNNs) have been successful in learning more complex functions than shallow ones (Bengio, 2009) and exceled in many challenging tasks such as in speech (Hinton et al., 2012) and vision (Krizhevsky et al., 2012). These results have sparked interest in applying DNNs to natural language processing problems as well. Specifically, in machine translation (MT), there has been an active body of work recently in utilizing neural language models (NLMs) to improve translation quality. However, to the best of our knowledge, work in this direction only makes use of NLMs with either one or two hidden layers. For example, Schwenk (2010) and Son et al. (2012) use 1-hidden-layer NLMs for reranking.[2] Vaswani et al. (2013) considered NLMs with two hidden layers for decoding but provided no comparison among models of various depths. Schwenk et al. (2012) and Devlin et al. (2014) reported only a small gain with 2-hidden-layer NLMs over single-layer ones.[3] There have not been clear results on whether adding more layers to NLMs helps.

In this paper, we demonstrate that deep NLMs with three or four layers are better than those with fewer layers in terms of the perplexity and the translation quality. We detail how we combine various techniques from past work to successfully train deep NLMs that condition on both the source and target contexts. When reranking n-best lists of a strong web-forum MT baseline, our deep models achieve an additional improvement of 0.5 TER / 0.5 BLEU compared to using a shallow NLM. Furthermore, by fine-tuning general in-domain NLMs with out-of-domain data, we obtain a similar boost of 1.0 TER / 0.5 BLEU points over a strong domain-adapted sms-chat baseline compared to utilizing a shallow NLM.

2 Neural Language Models

We briefly describe the NLM architecture and training objective used in this work as well as com-

[1] Our code and related materials are publicly available at http://stanford.edu/~lmthang/nlm.

[2] Schwenk (2010) builds language models while Son et al. (2012) forms translation models.

[3] Schwenk et al. (2012) constructs language models for reranking while Devlin et al. (2014) creates translation models for both reranking and decoding.

pare our approach to other related work.

Architecture. Neural language models are fundamentally feed-forward networks as described in (Bengio et al., 2003), but not necessarily limited to only a single hidden layer. Like any other language model, NLMs specify a distribution, $p(w|c)$, to predict the next word w given a context c. The first step is to lookup embeddings for words in the context and concatenate them to form an input, $h^{(0)}$, to the first hidden layer. We then repeatedly build up hidden representations as follows, for $l = 1, \ldots, n$:

$$h^{(l)} = f\left(W^{(l)} h^{(l-1)} + b^{(l)}\right) \quad (1)$$

where f is a non-linear fuction such as $\tanh$. The predictive distribution, $p(w|c)$, is then derived using the standard softmax:

$$\begin{aligned} \boldsymbol{s} &= W^{(sm)} h^{(n)} + b^{(sm)} \\ p(w|c) &= \frac{\exp(\boldsymbol{s}_w)}{\sum_{w \in V} \exp(\boldsymbol{s}_w)} \end{aligned} \quad (2)$$

Objective. The typical way of training NLMs is to maximize the training data likelihood, or equivalently, to minimize the cross-entropy objective of the following form: $\sum_{(c,w) \in T} -\log p(w|c)$.

Training NLMs can be prohibitively slow due to the computationally expensive softmax layer. As a result, past works have tried to use a more efficient version of the softmax such as the hierarchical softmax (Morin, 2005; Mnih and Hinton, 2007; Mnih and Hinton, 2009) or the class-based one (Mikolov et al., 2010; Mikolov et al., 2011). Recently, the noise-contrastive estimation (NCE) technique (Gutmann and Hyvärinen, 2012) has been applied to train NLMs in (Mnih and Teh, 2012; Vaswani et al., 2013) to avoid explicitly computing the normalization factors.

Devlin et al. (2014) used a modified version of the cross-entropy objective, the *self-normalized* one. The idea is to not only improve the prediction, $p(w|c)$, but also to push the normalization factor per context, Z_c, close to 1:

$$\mathbb{J} = \sum_{(c,w) \in T} -\log p(w|c) + \alpha \log^2(Z_c) \quad (3)$$

While self-normalization does not lead to speed up in training, it allows trained models to be applied efficiently at test time without computing the normalization factors. This is similar in flavor to NCE but allows for flexibility (through α) in how hard we want to "squeeze" the normalization factors.

Training deep NLMs. We follow (Devlin et al., 2014) to train self-normalized NLMs, conditioning on both the source and target contexts. Unlike (Devlin et al., 2014), we found that using the rectified linear function, $\max\{0, x\}$, proposed in (Nair and Hinton, 2010), works better than $\tanh$. The rectified linear function was used in (Vaswani et al., 2013) as well. Furthermore, while these works use a fixed learning rate throughout, we found that having a simple learning rate schedule is useful in training well-performing deep NLMs. This has also been demonstrated in (Sutskever et al., 2014; Luong et al., 2015) and is detailed in Section 3. We do not perform any gradient clipping and notice that learning is more stable when short sentences of length less than or equal to 2 are removed. Bias terms are used for all hidden layers as well as the softmax layer as described earlier, which is slightly different from other work such as (Vaswani et al., 2013). All these details contribute to our success in training deep NLMs.

For simplicity, the same vocabulary is used for both the embedding and the softmax matrices.[4] In addition, we adopt the standard softmax to take advantage of GPUs in performing large matrix multiplications. All hyperparameters are given later.

3 Experiments

3.1 Data

We use the Chinese-English bitext in the DARPA BOLT (Broad Operational Language Translation) program, with 11.1M parallel sentences (281M Chinese words and 307M English words). We reserve 585 sentences for validation, i.e., choosing hyperparameters, and 1124 sentences for testing.[5]

3.2 NLM Training

We train our NLMs described in Section 2 with SGD, using: (a) a source window of size 5, i.e., 11-gram source context[6], (b) a 4-word target history, i.e., 5-gram target LM, (c) a self-normalized weight $\alpha = 0.1$, (d) a mini-batch of size 128, and (e) a learning rate of 0.1 (training costs are normalized by the mini-batch size). All weights are uniformly initialized in $[-0.01, 0.01]$. We train

[4] Some work (Schwenk, 2010; Schwenk et al., 2012) utilize a smaller softmax vocabulary, called short-list.

[5] The test set is from BOLT and labeled as p1r6_dev.

[6] We used an alignment heuristic similar to Devlin et al. (2014) but applicable to our phrase-based MT system.

Models	Valid	Test	$\lvert \log Z \rvert$
1 layer	9.39	8.99	0.51
2 layers	9.20	8.96	0.50
3 layers	8.64	8.13	0.43
4 layers	**8.10**	**7.71**	**0.35**

Table 1: **Training NLMs** – validation and test perplexities achieved by self-normalized NLMs of various depths. We report the $\lvert \log Z \rvert$ value (base e), similar to Devlin et al. (2014), to indicate how good each model is in pushing the log normalization factors towards 0. All perplexities are derived by explicitly computing the normalization factors.

our models for 4 epochs (after 2 epochs, the learning rate is halved every 0.5 epoch). The vocabularies are limited to the top 40K frequent words for both Chinese and English. All words not in these vocabularies are replaced by a universal unknown symbol. Embeddings are of size 256 and all hidden layers have 512 units each. Our training speed on a single Tesla K40 GPU device is about 1000 target words per second and it generally takes about 10-14 days to fully train a model.

We present the NLM training results in Table 1. With more layers, the model succeeds in learning more complex functions; the prediction, hence, becomes more accurate as evidenced by smaller perplexities for both the validation and test sets. Interestingly, we observe that deeper nets can learn self-normalized NLMs better: the mean log normalization factor, $\lvert \log Z \rvert$ in Eq. (3), is driven towards 0 as the depth increases.[7]

3.3 MT Reranking with NLMs

Our MT models are built using the Phrasal MT toolkit (Cer et al., 2010). In addition to the standard dense feature set[8], we include a variety of sparse features for rules, word pairs, and word classes, as described in (Green et al., 2014). Our decoder uses three language models.[9] We use a tuning set of 396K words in the newswire and web domains and tune our systems using online expected error rate training as in (Green et al., 2014).

[7] As a reference point, though not directly comparable, Devlin et al. (2014) achieved 0.68 for $\lvert \log Z \rvert$ on a different test set with the same self-normalized constant $\alpha = 0.1$.

[8] Consisting of forward and backward translation models, lexical weighting, linear distortion, word penalty, phrase penalty and language model.

[9] One is trained on the English side of the bitext, one is trained on a 16.3-billion-word monolingual corpus taken from various domains, and one is a class-based language model trained on the same large monolingual corpus.

System	*dev*		*test1*		*test2*	
	T↓	B↑	T↓	B↑	T↓	B↑
baseline	53.7	33.1	55.1	31.3	63.5	16.5
Reranking						
1 layer	52.9	34.3	54.5	32.0	63.1	16.7
2 layers	52.7	34.1	54.3	31.9	63.0	16.8
3 layers	**52.5**	34.5	54.3	32.3	**62.5**	**17.3**
4 layers	52.6	**34.7**	**54.1**	**32.4**	62.7	17.2
vs. baseline	+1.2†	+1.6†	+1.0†	+1.1†	+1.0†	+0.8†
vs. 1 layer	+0.4†	+0.4†	+0.4†	+0.4†	+0.6†	+0.6†

Table 2: **Web-forum Results** – TER (T) and BLEU (B) scores on both the dev set (dev10wb_dev), used to tune reranking weights, and the test sets (dev10wb_syscomtune and p1r6_dev accordingly). Relative improvements between the best system and the baseline as well as the 1-layer model are **bolded**. † marks improvements that are statistically significant ($p < 0.05$).

Our tuning metric is (BLEU-TER)/2.

We run a discriminative reranker on the 1000-best output of a decoder with MERT. The features used in reranking include all the dense features, an aggregate decoder score, and an NLM score. We learn the reranker weights on a second tuning set, different from the decoder tuning set, to make the reranker less biased towards the dense features. This second tuning set consists of 33K words of web-forum text and is important to obtain good improvements with reranking.

3.4 Results

As shown in Table 2, it is not obvious if the depth-2 model is better than the single layer one, both of which are what past work used. In contrast, reranking with deep NLMs of three or four layers are clearly better, yielding average improvements of 1.0 TER / 1.0 BLEU points over the baseline and 0.5 TER / 0.5 BLEU points over the system reranked with the 1-layer model, all of which are statisfically significant according to the test described in (Riezler and Maxwell, 2005).

The fact that the improvements in terms of the intrinsic metrics listed in Table 1 do translate into gains in translation quality is interesting. It reinforces the trend reported in (Luong et al., 2015) that better source-conditioned perplexities lead to better translation scores. This phenomon is a useful result as in the past, many intrinsic metrics, e.g., alignment error rate, do not necessarily correlate with MT quality metrics.

System		dev		test	
		T↓	B↑	T↓	B↑
baseline		62.2	18.7	57.3	23.3
Reranking					
1 layer	*(5.42, 0.51)*	60.1	**22.0**	56.2	25.9
2 layers	*(5.50, 0.51)*	60.7	21.5	56.3	26.0
3 layers	*(5.34 0.43)*	**59.9**	21.4	**55.2**	**26.4**
vs. baseline		+2.3‡	+3.3‡	+2.1‡	+3.1‡
vs. 1 layer		+0.2	-0.6	+1.0‡	+0.5

Table 3: **Domain-adaptation Results** – translation scores for the sms-chat domain similar to Table 2. We use p2r2smscht_dev for *dev* and p2r2smscht_syscomtune for *test*. The test perplexities and the $|\log Z|$ values of our domain-adapted NLMs are shown in *italics*. ‡ marks improvements that are statistically significant ($p<0.01$).

3.5 Domain Adaptation

For the sms-chat domain, we use a tune set of 260K words in the newswire, web, and sms-chat domains to tune the decoder weights and a separate small, 8K words set to tune reranking weights. To train adapted NLMs, we use models previously trained on general in-domain data and further fine-tune with out-domain data for about 4 hours.[10]

Similar to the web-forum domain, for sms-chat, Table 3 shows that on the test set, our deep NLM with three layers yields a significant gain of 2.1 TER / 3.1 BLEU points over the baseline and 1.0 TER / 0.5 BLEU points over the 1-layer reranked system. It is worth pointing out that for such a small amount of out-domain training data, depth becomes less effective as exhibited through the insignificant BLEU gain in *test* and a drop in *dev* when comparing between the 1- and 3-layer models. We exclude the 4-layer NLM as it seems to have overfitted the training data. Nevertheless, we still achieve decent gains in using NLMs for MT domain adaptation.

4 Analysis

4.1 NLM Training

We show in Figure 1 the learning curves for various NLMs, demonstrating that deep nets are better than the shallow NLM with a single hidden layer. Starting from minibatch 20K, the ranking is generally maintained that deeper NLMs have better

[10]Our sms-chat corpus consists of 146K sentences (1.6M Chinese and 1.9M English words). We randomly select 3000 sentences for validation and 3000 sentences for test. Models are trained for 8 iterations with the same hyperparameters.

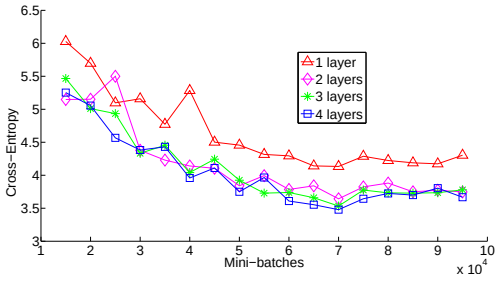

Figure 1: **NLM Learning Curve** – test cross-entropies ($\log_e$ perplexities) for various NLMs.

cross-entropies. The gaps become less discernible from minibatch 30K onwards, but numerically, as the model becomes deeper, the average gaps, in perplexities, are consistently 40.1, 1.1, and 2.0.

4.2 Reranking Settings

In Table 4, we compare reranking using all dense features (*All*) to conditions which use only dense LM features (*LM*) and optionally, include a word penalty (WP) feature. All these settings include an NLM score and an aggregate decoder score. As shown, it is best to include *all* dense features at reranking time.

	All	LMs + WP	LMs
1 layer	11.3	11.3	11.4
2 layers	11.2	11.4	11.5
3 layers	11.0	11.1	11.4
4 layers	10.9	11.2	11.3

Table 4: **Reranking Conditions** – (TER-BLEU)/2 scores when reranking the web-forum baseline.

5 Related Work

It is worth mentioning another active line of research in building end-to-end neural MT systems (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015; Jean et al., 2015). These methods have not yet demonstrated success on challenging language pairs such as English-Chinese. Arsoy et al. (2012) have preliminarily examined deep NLMs for speech recognition, however, we believe, this is the first work that puts deep NLMs into the context of MT.

6 Conclusion

In this paper, we have bridged the gap that past work did not show, that is, neural language models with more than two layers can help improve

translation quality. Our results confirm the trend reported in (Luong et al., 2015) that source-conditioned perplexity strongly correlates with MT performance. We have also demonstrated the use of deep NLMs to obtain decent gains in out-of-domain conditions.

Acknowledgment

We gratefully acknowledge support from a gift from Bloomberg L.P. and from the Defense Advanced Research Projects Agency (DARPA) Broad Operational Language Translation (BOLT) program under contract HR0011-12-C-0015 through IBM. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of DARPA, or the US government. We thank members of the Stanford NLP Group as well as the annonymous reviewers for their valuable comments and feedbacks.

References

Ebru Arsoy, Tara N. Sainath, Brian Kingsbury, and Bhuvana Ramabhadran. 2012. Deep neural network language models. In *NAACL WLM Workshop*.

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. 2003. A neural probabilistic language model. *JMLR*, 3:1137–1155.

Yoshua Bengio. 2009. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127, January.

D. Cer, M. Galley, D. Jurafsky, and C. D. Manning. 2010. Phrasal: A statistical machine translation toolkit for exploring new model features. In *ACL, Demonstration Session*.

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

S. Green, D. Cer, and C. D. Manning. 2014. An empirical comparison of features and tuning for phrase-based machine translation. In *WMT*.

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

G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury. 2012. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *ACL*.

N. Kalchbrenner and P. Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *NIPS*.

M.-T. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *ACL*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*.

Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *ICASSP*.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *ICML*.

Andriy Mnih and Geoffrey Hinton. 2009. A scalable hierarchical distributed language model. In *NIPS*.

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

Frederic Morin. 2005. Hierarchical probabilistic neural network language model. In *AISTATS*.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*.

Stefan Riezler and T. John Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for MT. In *ACL Workshop, Intrinsic/Extrinsic Evaluation Measures for MT and Summarization*.

H. Schwenk, A. Rousseau, and M. Attik. 2012. Large, pruned or continuous space language models on a gpu for statistical machine translation. In *NAACL WLM workshop*.

H. Schwenk. 2010. Continuous space language models for statistical machine translation. *The Prague Bulletin of Mathematical Linguistics*, (93):137–146.

Le Hai Son, Alexandre Allauzen, and François Yvon. 2012. Continuous space translation models with neural networks. In *NAACL-HLT*.

I. Sutskever, O. Vinyals, and Q. V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *EMNLP*.