Decoding with Large-Scale Neural Language Models Improves Translation

Ashish Vaswani
University of Southern California
Department of Computer Science
avaswani@isi.edu

Yinggong Zhao
Nanjing University, State Key Laboratory
for Novel Software Technology
zhaoyg@nlp.nju.edu.cn

Victoria Fossum and **David Chiang**
University of Southern California
Information Sciences Institute
{vfossum,chiang}@isi.edu

Abstract

We explore the application of neural language models to machine translation. We develop a new model that combines the neural probabilistic language model of Bengio et al., rectified linear units, and noise-contrastive estimation, and we incorporate it into a machine translation system both by reranking k-best lists and by direct integration into the decoder. Our large-scale, large-vocabulary experiments across four language pairs show that our neural language model improves translation quality by up to 1.1 Bleu.

1 Introduction

Machine translation (MT) systems rely upon language models (LMs) during decoding to ensure fluent output in the target language. Typically, these LMs are n-gram models over discrete representations of words. Such models are susceptible to data sparsity–that is, the probability of an n-gram observed only few times is difficult to estimate reliably, because these models do not use any information about similarities between words.

To address this issue, Bengio et al. (2003) propose distributed word representations, in which each word is represented as a real-valued vector in a high-dimensional feature space. Bengio et al. (2003) introduce a feed-forward neural probabilistic LM (NPLM) that operates over these distributed representations. During training, the NPLM learns both a distributed representation for each word in the vocabulary and an n-gram probability distribution over words in terms of these distributed representations.

Although neural LMs have begun to rival or even surpass traditional n-gram LMs (Mnih and Hinton, 2009; Mikolov et al., 2011), they have not yet been widely adopted in large-vocabulary applications such as MT, because standard maximum likelihood estimation (MLE) requires repeated summations over all words in the vocabulary. A variety of strategies have been proposed to combat this issue, many of which require severe restrictions on the size of the network or the size of the data.

In this work, we extend the NPLM of Bengio et al. (2003) in two ways. First, we use rectified linear units (Nair and Hinton, 2010), whose activations are cheaper to compute than sigmoid or tanh units. There is also evidence that deep neural networks with rectified linear units can be trained successfully without pre-training (Zeiler et al., 2013). Second, we train using noise-contrastive estimation or NCE (Gutmann and Hyvärinen, 2010; Mnih and Teh, 2012), which does not require repeated summations over the whole vocabulary. This enables us to efficiently build NPLMs on a larger scale than would be possible otherwise.

We then apply this LM to MT in two ways. First, we use it to rerank the k-best output of a hierarchical phrase-based decoder (Chiang, 2007). Second, we integrate it directly into the decoder, allowing the neural LM to more strongly influence the model. We achieve gains of up to 0.6 Bleu translating French, German, and Spanish to English, and up to 1.1 Bleu on Chinese-English translation.

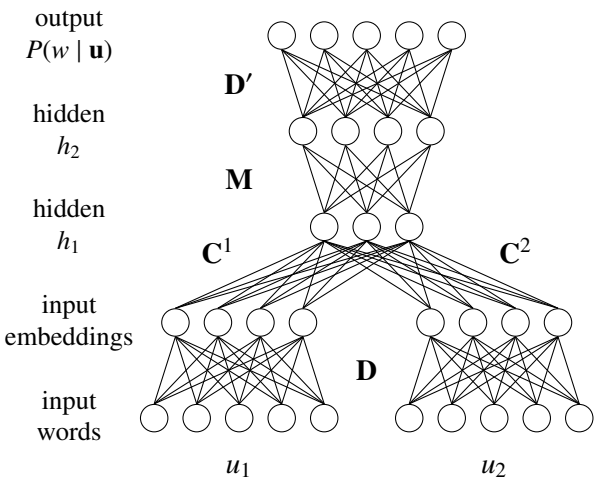


Figure 1: Neural probabilistic language model (Bengio et al., 2003).

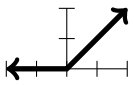
2 Neural Language Models

Let V be the vocabulary, and n be the order of the language model; let $\mathbf{u}$ range over contexts, i.e., strings of length $(n-1)$, and w range over words. For simplicity, we assume that the training data is a single very long string, $w_1 \cdots w_N$, where w_N is a special stop symbol, </s>. We write $\mathbf{u}_i$ for $w_{i-n+1} \cdots w_{i-1}$, where, for $i \leq 0$, w_i is a special start symbol, <s>.

2.1 Model

We use a feedforward neural network as shown in Figure 1, following Bengio et al. (2003). The input to the network is a sequence of one-hot representations of the words in context $\mathbf{u}$, which we write u_j $(1 \leq j \leq n-1)$. The output is the probability $P(w \mid \mathbf{u})$ for each word w, which the network computes as follows.

The hidden layers consist of rectified linear units (Nair and Hinton, 2010), which use the activation function $\phi(x) = \max(0, x)$ (see graph at right).



The output of the first hidden layer h_1 is

$$h_1 = \phi\left(\sum_{j=1}^{n-1} \mathbf{C}^j \mathbf{D} u_j\right) \tag{1}$$

where $\mathbf{D}$ is a matrix of input word embeddings which is shared across all positions, the $\mathbf{C}^j$ are the context matrices for each word in $\mathbf{u}$, and ϕ is applied elementwise. The output of the second layer h_2 is

$$h_2 = \phi\left(\mathbf{M} h_1\right),$$

where $\mathbf{M}$ is the matrix of connection weights between h_1 and h_2. Finally, the output layer is a softmax layer,

$$P(w \mid \mathbf{u}) \propto \exp\left(\mathbf{D}' h_2 + \mathbf{b}\right) \tag{2}$$

where $\mathbf{D}'$ is the output word embedding matrix and $\mathbf{b}$ is a vector of biases for every word in the vocabulary.

2.2 Training

The typical way to train neural LMs is to maximize the likelihood of the training data by gradient ascent. But the softmax layer requires, at each iteration, a summation over all the units in the output layer, that is, all words in the whole vocabulary. If the vocabulary is large, this can be prohibitively expensive.

Noise-contrastive estimation or NCE (Gutmann and Hyvärinen, 2010) is an alternative estimation principle that allows one to avoid these repeated summations. It has been applied previously to log-bilinear LMs (Mnih and Teh, 2012), and we apply it here to the NPLM described above.

We can write the probability of a word w given a context $\mathbf{u}$ under the NPLM as

$$\begin{aligned} P(w \mid \mathbf{u}) &= \frac{1}{Z(\mathbf{u})} p(w \mid \mathbf{u}) \\ p(w \mid \mathbf{u}) &= \exp\left(\mathbf{D}' h_2 + \mathbf{b}\right) \\ Z(\mathbf{u}) &= \sum_{w'} p(w' \mid \mathbf{u}) \end{aligned} \tag{3}$$

where $p(w \mid \mathbf{u})$ is the *unnormalized* output of the unit corresponding to w, and $Z(\mathbf{u})$ is the normalization factor. Let θ stand for the parameters of the model.

One possibility would be to treat $Z(\mathbf{u})$, instead of being defined by (3), as an additional set of model parameters which are learned along with θ. But it is easy to see that we can make the likelihood arbitrarily large by making the $Z(\mathbf{u})$ arbitrarily small.

In NCE, we create a noise distribution $q(w)$. For each example $\mathbf{u}_i w_i$, we add k noise samples $\bar{w}_{i1}, \ldots, \bar{w}_{ik}$ into the data, and extend the model to account for noise samples by introducing a random

variable C which is 1 for training examples and 0 for noise samples:

$$P(C = 1, w \mid \mathbf{u}) = \frac{1}{1+k} \cdot \frac{1}{Z(\mathbf{u})} p(w \mid \mathbf{u})$$
$$P(C = 0, w \mid \mathbf{u}) = \frac{k}{1+k} \cdot q(w).$$

We then train the model to classify examples as training data or noise, that is, to maximize the *conditional* likelihood,

$$L = \sum_{i=1}^{N} \Bigg(\log P(C = 1 \mid \mathbf{u}_i w_i) + \sum_{j=1}^{k} \log P(C = 0 \mid \mathbf{u}_i \bar{w}_{ij}) \Bigg)$$

with respect to both θ and $Z(\mathbf{u})$.

We do this by stochastic gradient ascent. The gradient with respect to θ turns out to be

$$\frac{\partial L}{\partial \theta} = \sum_{i=1}^{N} \Bigg(P(C = 0 \mid \mathbf{u}_i w_i) \frac{\partial}{\partial \theta} \log p(w_i \mid \mathbf{u}_i) - \sum_{j=1}^{k} P(C = 1 \mid \mathbf{u}_i \bar{w}_{ij}) \frac{\partial}{\partial \theta} \log p(\bar{w}_{ij} \mid \mathbf{u}_i) \Bigg)$$

and similarly for the gradient with respect to $Z(\mathbf{u})$. These can be computed by backpropagation. Unlike before, the $Z(\mathbf{u})$ will converge to a value that normalizes the model, satisfying (3), and, under appropriate conditions, the parameters will converge to a value that maximizes the likelihood of the data.

3 Implementation

Both training and scoring of neural LMs are computationally expensive at the scale needed for machine translation. In this section, we describe some of the techniques used to make them practical for translation.

3.1 Training

During training, we compute gradients on an entire minibatch at a time, allowing the use of matrix-matrix multiplications instead of matrix-vector multiplications (Bengio, 2012). We represent the inputs as a sparse matrix, allowing the computation of the input layer (1) to use sparse matrix-matrix multiplications. The output activations (2) are computed only for the word types that occur as the positive example or one of the noise samples, yielding a sparse matrix of outputs. Similarly, during backpropagation, sparse matrix multiplications are used at both the output and input layer.

In most of these operations, the examples in a minibatch can be processed in parallel. However, in the sparse-dense products used when updating the parameters $\mathbf{D}$ and $\mathbf{D}'$, we found it was best to divide the vocabulary into blocks (16 per thread) and to process the blocks in parallel.

3.2 Translation

To incorporate this neural LM into a MT system, we can use the LM to rerank k-best lists, as has been done in previous work. But since the NPLM scores n-grams, it can also be integrated into a phrase-based or hierarchical phrase-based decoder just as a conventional n-gram model can, unlike a RNN.

The most time-consuming step in computing n-gram probabilities is the computation of the normalization constants $Z(\mathbf{u})$. Following Mnih and Teh (2012), we set all the normalization constants to one during training, so that the model learns to produce approximately normalized probabilities. Then, when applying the LM, we can simply ignore normalization. A similar strategy was taken by Niehues and Waibel (2012). We find that a single n-gram lookup takes about 40 μs.

The technique, described above, of grouping examples into minibatches works for scoring of k-best lists, but not while decoding. But caching n-gram probabilities helps to reduce the cost of the many lookups required during decoding.

A final issue when decoding with a neural LM is that, in order to estimate future costs, we need to be able to estimate probabilities of n'-grams for $n' < n$. In conventional LMs, this information is readily available,[1] but not in NPLMs. Therefore, we defined a special word <null> whose embedding is the weighted average of the (input) embeddings of all the other words in the vocabulary. Then, to estimate the probability of an n'-gram $\mathbf{u}'w$, we used the probability of $P(w \mid \texttt{<null>}^{n-n'}\mathbf{u}')$.

[1] However, in Kneser-Ney smoothed LMs, this information is also incorrect (Heafield et al., 2012).

setting	dev	2004	2005	2006
baseline	38.2	38.4	37.7	34.3
reranking	38.5	38.6	37.8	34.7
decoding	39.1	39.5	38.8	34.9

Table 1: Results for Chinese-English experiments, without neural LM (baseline) and with neural LM for reranking and integrated decoding. Reranking with the neural LM improves translation quality, while integrating it into the decoder improves even more.

	Fr-En		De-En		Es-En	
setting	dev	test	dev	test	dev	test
baseline	33.5	25.5	28.8	21.5	33.5	32.0
reranking	33.9	26.0	29.1	21.5	34.1	32.2
decoding	34.1[2]	26.1[2]	29.3	21.9	34.2[2]	32.1[2]

Table 2: Results for Europarl MT experiments, without neural LM (baseline) and with neural LM for reranking and integrated decoding. The neural LM gives improvements across three different language pairs. Superscript 2 indicates a score averaged between two runs; all other scores were averaged over three runs.

4 Experiments

We ran experiments on four language pairs – Chinese to English and French, German, and Spanish to English – using a hierarchical phrase-based MT system (Chiang, 2007) and GIZA++ (Och and Ney, 2003) for word alignments.

For all experiments, we used four LMs. The baselines used conventional 5-gram LMs, estimated with modified Kneser-Ney smoothing (Chen and Goodman, 1998) on the English side of the bitext and the 329M-word Xinhua portion of English Gigaword (LDC2011T07). Against these baselines, we tested systems that included the two conventional LMs as well as two 5-gram NPLMs trained on the same datasets. The Europarl bitext NPLMs had a vocabulary size of 50k, while the other NPLMs had a vocabulary size of 100k. We used 150 dimensions for word embeddings, 750 units in hidden layer h_1, and 150 units in hidden layer h_2. We initialized the network parameters uniformly from $(-0.01, 0.01)$ and the output biases to $-\log|V|$, and optimized them by 10 epochs of stochastic gradient ascent, using minibatches of size 1000 and a learning rate of 1. We drew 100 noise samples per training example from the unigram distribution, using the alias method for efficiency (Kronmal and Peterson, 1979).

We trained the discriminative models with MERT (Och, 2003) and the discriminative rerankers on 1000-best lists with MERT. Except where noted, we ran MERT three times and report the average score. We evaluated using case-insensitive NIST BLEU.

4.1 NIST Chinese-English

For the Chinese-English task (Table 1), the training data came from the NIST 2012 constrained track, excluding sentences longer than 60 words. Rules without nonterminals were extracted from all training data, while rules with nonterminals were extracted from the FBIS corpus (LDC2003E14). We ran MERT on the development data, which was the NIST 2003 test data, and tested on the NIST 2004–2006 test data.

Reranking using the neural LM yielded improvements of 0.2–0.4 BLEU, while integrating the neural LM yielded larger improvements, between 0.6 and 1.1 BLEU.

4.2 Europarl

For French, German, and Spanish translation, we used a parallel text of about 50M words from Europarl v7. Rules without nonterminals were extracted from all the data, while rules with nonterminals were extracted from the first 200k words. We ran MERT on the development data, which was the WMT 2005 test data, and tested on the WMT 2006 news commentary test data (nc-test2006).

The improvements, shown in Table 2, were more modest than on Chinese-English. Reranking with the neural LM yielded improvements of up to 0.5 BLEU, and integrating the neural LM into the decoder yielded improvements of up to 0.6 BLEU. In one case (Spanish-English), integrated decoding scored higher than reranking on the development data but lower on the test data – perhaps due to the difference in domain between the two. On the other tasks, integrated decoding outperformed reranking.

4.3 Speed comparison

We measured the speed of training a NPLM by NCE, compared with MLE as implemented by the CSLM toolkit (Schwenk, 2013). We used the first 200k

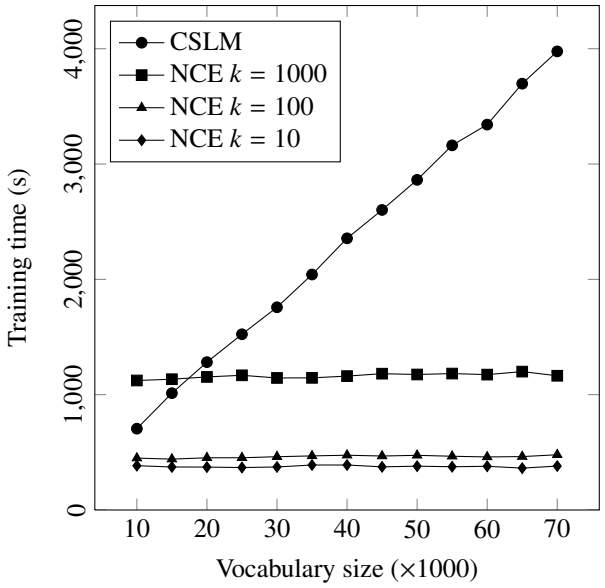


Figure 2: Noise contrastive estimation (NCE) is much faster, and much less dependent on vocabulary size, than MLE as implemented by the CSLM toolkit (Schwenk, 2013).

lines (5.2M words) of the Xinhua portion of Gigaword and timed one epoch of training, for various values of k and $|V|$, on a dual hex-core 2.67 GHz Xeon X5650 machine. For these experiments, we used minibatches of 128 examples. The timings are plotted in Figure 2. We see that NCE is considerably faster than MLE; moreover, as expected, the MLE training time is roughly linear in $|V|$, whereas the NCE training time is basically constant.

5 Related Work

The problem of training with large vocabularies in NPLMs has received much attention. One strategy has been to restructure the network to be more hierarchical (Morin and Bengio, 2005; Mnih and Hinton, 2009) or to group words into classes (Le et al., 2011). Other strategies include restricting the vocabulary of the NPLM to a *shortlist* and reverting to a traditional n-gram LM for other words (Schwenk, 2004), and limiting the number of training examples using resampling (Schwenk and Gauvain, 2005) or selecting a subset of the training data (Schwenk et al., 2012). Our approach can be efficiently applied to large-scale tasks without limiting either the model or the data.

NPLMs have previously been applied to MT, most notably feed-forward NPLMs (Schwenk, 2007; Schwenk, 2010) and RNN-LMs (Mikolov, 2012). However, their use in MT has largely been limited to reranking k-best lists for MT tasks with restricted vocabularies. Niehues and Waibel (2012) integrate a RBM-based language model directly into a decoder, but they only train the RBM LM on a small amount of data. To our knowledge, our approach is the first to integrate a large-vocabulary NPLM directly into a decoder for a large-scale MT task.

6 Conclusion

We introduced a new variant of NPLMs that combines the network architecture of Bengio et al. (2003), rectified linear units (Nair and Hinton, 2010), and noise-contrastive estimation (Gutmann and Hyvärinen, 2010). This model is dramatically faster to train than previous neural LMs, and can be trained on a large corpus with a large vocabulary and directly integrated into the decoder of a MT system. Our experiments across four language pairs demonstrated improvements of up to 1.1 BLEU. Code for training and using our NPLMs is available for download.[2]

Acknowledgements

We would like to thank the anonymous reviewers for their very helpful comments. This research was supported in part by DOI IBC grant D12AP00225. This work was done while the second author was visiting USC/ISI supported by China Scholarship Council. He was also supported by the Research Fund for the Doctoral Program of Higher Education of China (No. 20110091110003) and the National Fundamental Research Program of China (2010CB327903).

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*.

Yoshua Bengio. 2012. Practical recommendations for gradient-based training of deep architectures. *CoRR*, abs/1206.5533.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language mod-

[2] `http://nlg.isi.edu/software/nplm`

eling. Technical Report TR-10-98, Harvard University Center for Research in Computing Technology.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of AISTATS*.

Kenneth Heafield, Philipp Koehn, and Alon Lavie. 2012. Language model rest costs and space-efficient storage. In *Proceedings of EMNLP-CoNLL*, pages 1169–1178.

Richard Kronmal and Arthur Peterson. 1979. On the alias method for generating random variables from a discrete distribution. *The American Statistician*, 33(4):214–218.

Hai-Son Le, Ilya Oparin, Alexandre Allauzen, Jean-Luc Gauvain, and François Yvon. 2011. Structured output layer neural network language model. In *Proceedings of ICASSP*, pages 5524–5527.

Tomáš Mikolov, Anoop Deoras, Stefan Kombrink, Lukáš Burget, and Jan "Honza" Černocký. 2011. Empirical evaluation and combination of advanced language modeling techniques. In *Proceedings of INTERSPEECH*, pages 605–608.

Tomáš Mikolov. 2012. *Statistical Language Models Based on Neural Networks*. Ph.D. thesis, Brno University of Technology.

Andriy Mnih and Geoffrey Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems*.

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

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of AISTATS*, pages 246–252.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of ICML*, pages 807–814.

Jan Niehues and Alex Waibel. 2012. Continuous space language models using Restricted Boltzmann Machines. In *Proceedings of IWSLT*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167.

Holger Schwenk and Jean-Luc Gauvain. 2005. Training neural network language models on very large corpora. In *Proceedings of EMNLP*.

Holger Schwenk, Anthony Rousseau, and Mohammed Attik. 2012. Large, pruned or continuous space language models on a GPU for statistical machine translation. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 11–19.

Holger Schwenk. 2004. Efficient training of large neural networks for language modeling. In *Proceedings of IJCNN*, pages 3059–3062.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech and Language*, 21:492–518.

Holger Schwenk. 2010. Continuous-space language models for statistical machine translation. *Prague Bulletin of Mathematical Linguistics*, 93:137–146.

Holger Schwenk. 2013. CSLM - a modular open-source continuous space language modeling toolkit. In *Proceedings of Interspeech*.

M.D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q.V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G.E. Hinton. 2013. On rectified linear units for speech processing. In *Proceedings of ICASSP*.