Decoder Integration and Expected BLEU Training for Recurrent Neural Network Language Models

Michael Auli
Microsoft Research
Redmond, WA, USA
michael.auli@microsoft.com

Jianfeng Gao
Microsoft Research
Redmond, WA, USA
jfgao@microsoft.com

Abstract

Neural network language models are often trained by optimizing likelihood, but we would prefer to optimize for a task specific metric, such as BLEU in machine translation. We show how a recurrent neural network language model can be optimized towards an expected BLEU loss instead of the usual cross-entropy criterion. Furthermore, we tackle the issue of directly integrating a recurrent network into first-pass decoding under an efficient approximation. Our best results improve a phrase-based statistical machine translation system trained on WMT 2012 French-English data by up to 2.0 BLEU, and the expected BLEU objective improves over a cross-entropy trained model by up to 0.6 BLEU in a single reference setup.

1 Introduction

Neural network-based language and translation models have achieved impressive accuracy improvements on statistical machine translation tasks (Allauzen et al., 2011; Le et al., 2012b; Schwenk et al., 2012; Vaswani et al., 2013; Gao et al., 2014). In this paper we focus on recurrent neural network architectures which have recently advanced the state of the art in language modeling (Mikolov et al., 2010; Mikolov et al., 2011; Sundermeyer et al., 2013) with several subsequent applications in machine translation (Auli et al., 2013; Kalchbrenner and Blunsom, 2013; Hu et al., 2014). Recurrent models have the potential to capture long-span dependencies since their predictions are based on an *unbounded history* of previous words (§2).

In practice, neural network models for machine translation are usually trained by maximizing the likelihood of the training data, either via a cross-entropy objective (Mikolov et al., 2010; Schwenk et al., 2012) or more recently, noise-contrastive estimation (Vaswani et al., 2013). However, it is widely appreciated that directly optimizing for a task-specific metric often leads to better performance (Goodman, 1996; Och, 2003; Auli and Lopez, 2011). The expected BLEU objective provides an efficient way of achieving this for machine translation (Rosti et al., 2010; Rosti et al., 2011; He and Deng, 2012; Gao and He, 2013; Gao et al., 2014) instead of solely relying on traditional optimizers such as Minimum Error Rate Training (MERT) that only adjust the weighting of entire component models within the log-linear framework of machine translation (§3).

Most previous work on neural networks for machine translation is based on a rescoring setup (Arisoy et al., 2012; Mikolov, 2012; Le et al., 2012a; Auli et al., 2013), thereby side stepping the algorithmic and engineering challenges of direct decoder-integration. One recent exception is Vaswani et al. (2013) who demonstrated that feed-forward network-based language models are more accurate in first-pass decoding than in rescoring. Decoder integration has the advantage for the neural network to directly influence search, unlike rescoring which is restricted to an n-best list or lattice. Decoding with feed-forward architectures is straightforward, since predictions are based on a fixed size input, similar to n-gram language models. However, for recurrent networks we have to deal with the *unbounded history*, which breaks the usual dynamic programming assumptions for efficient search. We show how a simple but effective approximation can side step this issue and we empirically demonstrate its effectiveness (§4).

We test the expected BLEU objective by training a recurrent neural network language model and obtain substantial improvements. We also find that our efficient approximation for decoder integration is very accurate, clearly outperforming a rescoring setup (§5).

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 136–142,
Baltimore, Maryland, USA, June 23-25 2014.

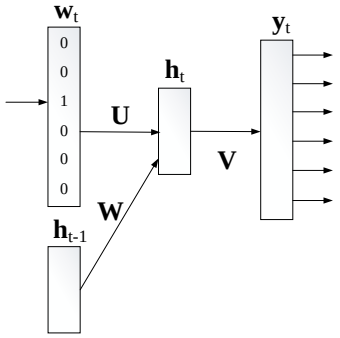

Figure 1: Structure of the recurrent neural network language model.

2 Recurrent Neural Network LMs

Our model has a similar structure to the recurrent neural network language model of Mikolov et al. (2010) which is factored into an input layer, a hidden layer with recurrent connections, and an output layer (Figure 1). The input layer encodes the word at position *t* as a 1-of-N vector $\mathbf{w}_t$. The output layer $\mathbf{y}_t$ represents scores over possible next words; both the input and output layers are of size $|V|$, the size of the vocabulary. The hidden layer state $\mathbf{h}_t$ encodes the history of all words observed in the sequence up to time step t. The state of the hidden layer is determined by the input layer and the hidden layer configuration of the previous time step $\mathbf{h}_{t-1}$. The weights of the connections between the layers are summarized in a number of matrices: $\mathbf{U}$ represents weights from the input layer to the hidden layer, and $\mathbf{W}$ represents connections from the previous hidden layer to the current hidden layer. Matrix $\mathbf{V}$ contains weights between the current hidden layer and the output layer. The activations of the hidden and output layers are computed by:

$$\mathbf{h}_t = \tanh(\mathbf{U}\mathbf{w}_t + \mathbf{W}\mathbf{h}_{t-1})$$
$$\mathbf{y}_t = \tanh(\mathbf{V}\mathbf{h}_t)$$

Different to previous work (Mikolov et al., 2010), we do not use the softmax activation function to output a probability over the next word, but instead just compute a *single unnormalized score*. This is computationally more efficient than summing over all possible outputs such as required for the cross-entropy error function (Bengio et al., 2003; Mikolov et al., 2010; Schwenk et al., 2012). Training is based on the back propagation through time algorithm, which unrolls the network and then computes error gradients over multiple time steps (Rumelhart et al., 1986); we use the expected BLEU loss (§3) to obtain the error with respect to the output activations. After training, the output layer represents scores $s(w_{t+1}|w_1 \dots w_t, \mathbf{h}_t)$ for the next word given the previous t input words and the current hidden layer configuration $\mathbf{h}_t$.

3 Expected BLEU Training

We integrate the recurrent neural network language model as an additional feature into the standard log-linear framework of translation (Och, 2003). Formally, our phrase-based model is parameterized by M parameters Λ where each $\lambda_m \in \Lambda$, $m = 1 \dots M$ is the weight of an associated feature $h_m(f, e)$. Function $h(f, e)$ maps foreign sentences f and English sentences e to the vector $h_1(f, e) \dots (f, e)$, and the model chooses translations according to the following decision rule:

$$\hat{e} = \arg\max_{e \in \mathcal{E}(f)} \Lambda^{\mathsf{T}} h(f, e)$$

We summarize the weights of the recurrent neural network language model as $\theta = \{\mathbf{U}, \mathbf{W}, \mathbf{V}\}$ and add the model as an additional feature to the log-linear translation model using the simplified notation $s_\theta(w_t) = s(w_t|w_1 \dots w_{t-1}, \mathbf{h}_{t-1})$:

$$h_{M+1}(e) = s_\theta(e) = \sum_{t=1}^{|e|} \log s_\theta(w_t) \qquad (1)$$

which computes a sentence-level language model score as the sum of individual word scores. The translation model is parameterized by Λ and θ which are learned as follows (Gao et al., 2014):

1. We generate an n-best list for each foreign sentence in the training data with the baseline translation system given Λ where $\lambda_{M+1} = 0$ using the settings described in §5. The n-best lists serve as an approximation to $\mathcal{E}(f)$ used in the next step for expected BLEU training of the recurrent neural network model (§3.1).

2. Next, we fix Λ, set $\lambda_{M+1} = 1$ and optimize θ with respect to the loss function on the training data using stochastic gradient descent (SGD).[1]

[1] We tuned λ_{M+1} on the development set but found that $\lambda_{M+1} = 1$ resulted in faster training and equal accuracy.

3. We fix θ and re-optimize Λ in the presence of the recurrent neural network model using Minimum Error Rate Training (Och, 2003) on the development set (§5).

3.1 Expected BLEU Objective

Formally, we define our loss function $l(\theta)$ as the negative expected BLEU score, denoted as xBLEU(θ) for a given foreign sentence f:

$$\begin{aligned} l(\theta) &= -\text{ xBLEU}(\theta) \\ &= \sum_{e \in \mathcal{E}(f)} p_{\Lambda,\theta}(e|f) \text{sBLEU}(e, e^{(i)}) \end{aligned} \quad (2)$$

where sBLEU$(e, e^{(i)})$ is a smoothed sentence-level BLEU score with respect to the reference translation $e^{(i)}$, and $\mathcal{E}(f)$ is the generation set given by an n-best list.[2] We use a sentence-level BLEU approximation similar to He and Deng (2012).[3] The normalized probability $p_{\Lambda,\theta}(e|f)$ of a particular translation e given f is defined as:

$$p_{\Lambda,\theta}(e|f) = \frac{\exp\{\gamma \Lambda^T h(f,e)\}}{\sum_{e' \in \mathcal{E}(f)} \exp\{\gamma \Lambda^T h(f,e')\}} \quad (3)$$

where $\Lambda^T h(f,e)$ includes the recurrent neural network $h_{M+1}(e)$, and $\gamma \in [0, \inf)$ is a scaling factor that flattens the distribution for $\gamma < 1$ and sharpens it for $\gamma > 1$ (Tromble et al., 2008).[4]

Next, we define the gradient of the expected BLEU loss function $l(\theta)$ using the observation that the loss does not explicitly depend on θ:

$$\begin{aligned} \frac{\partial l(\theta)}{\partial \theta} &= \sum_e \sum_{t=1}^{|e|} \frac{\partial l(\theta)}{\partial s_\theta(w_t)} \frac{\partial s_\theta(w_t)}{\partial \theta} \\ &= \sum_e \sum_{t=1}^{|e|} -\delta_{w_t} \frac{\partial s_\theta(w_t)}{\partial \theta} \end{aligned}$$

where δ_{w_t} is the *error term* for English word w_t.[5] The error term indicates how the loss changes with the translation probability which we derive next.[6]

[2] Our definitions do not take into account multiple derivations for the same translation because our n-best lists contain only unique entries which we obtain by choosing the highest scoring translation among string identical candidates.

[3] In early experiments we found that the BLEU+1 approximation used by Liang et al. (2006) and Nakov et. al (2012) worked equally well in our setting.

[4] The γ parameter is *only* used during expected BLEU training but not for subsequent MERT tuning.

[5] A sentence may contain the same word multiple times and we compute the error term for each occurrence separately since the error depends on the individual history.

[6] We omit the gradient of the recurrent neural network score $\frac{\partial s_\theta(w_t)}{\partial \theta}$ since it follows the standard form (Mikolov, 2012).

3.2 Derivation of the Error Term δ_{w_t}

We rewrite the loss function (2) using (3) and separate it into two terms $G(\theta)$ and $Z(\theta)$ as follows:

$$\begin{aligned} l(\theta) &= -\text{xBLEU}(\theta) = -\frac{G(\theta)}{Z(\theta)} \\ &= -\frac{\sum_{e \in \mathcal{E}(f)} \exp\{\gamma \Lambda^T h(f,e)\} \text{ sBLEU}(e, e^{(i)})}{\sum_{e \in \mathcal{E}(f)} \exp\{\gamma \Lambda^T h(f,e)\}} \end{aligned} \quad (4)$$

Next, we apply the quotient rule of differentiation:

$$\begin{aligned} \delta_{w_t} &= \frac{\partial \text{xBLEU}(\theta)}{\partial s_\theta(w_t)} = \frac{\partial (G(\theta)/Z(\theta))}{\partial s_\theta(w_t)} \\ &= \frac{1}{Z(\theta)} \left(\frac{\partial G(\theta)}{\partial s_\theta(w_t)} - \frac{\partial Z(\theta)}{\partial s_\theta(w_t)} \text{xBLEU}(\theta) \right) \end{aligned}$$

Using the observation that θ is only relevant to the recurrent neural network $h_{M+1}(e)$ (1) we have

$$\frac{\partial \gamma \Lambda^T h(f,e)}{\partial s_\theta(w_t)} = \gamma \lambda_{M+1} \frac{\partial h_{M+1}(e)}{\partial s_\theta(w_t)} = \frac{\gamma \lambda_{M+1}}{s_\theta(w_t)}$$

which together with the chain rule, (3) and (4) allows us to rewrite δ_{w_t} as follows:

$$\begin{aligned} \delta_{w_t} &= \frac{1}{Z(\theta)} \sum_{\substack{e \in \mathcal{E}(f), \\ s.t. w_t \in e}} \left(\frac{\partial \exp\{\gamma \Lambda^T h(f,e)\}}{\partial s_\theta(w_t)} U(\theta, e) \right) \\ &= \sum_{\substack{e \in \mathcal{E}(f), \\ s.t. w_t \in e}} \left(p_{\Lambda,\theta}(e|f) U(\theta, e) \lambda_{M+1} \frac{\gamma}{s_\theta(w_t)} \right) \end{aligned}$$

where $U(\theta, e) = \text{sBLEU}(e, e_i) - \text{xBLEU}(\theta)$.

4 Decoder Integration

Directly integrating our recurrent neural network language model into first-pass decoding enables us to search a much larger space than would be possible in rescoring.

Typically, phrase-based decoders maintain a set of *states* representing partial and complete translation hypothesis that are scored by a set of features. Most features are local, meaning that all required information for them to assign a score is available within the state. One exception is the n-gram language model which requires the preceding $n-1$ words as well. In order to accommodate this feature, each state usually keeps these words as *context*. Unfortunately, a recurrent neural network makes even weaker independence assumptions so

that it depends on the entire left prefix of a sentence. Furthermore, the weaker independence assumptions also dramatically reduce the effectiveness of dynamic programming by allowing much fewer states to be recombined.[7]

To solve this problem, we follow previous work on lattice rescoring with recurrent networks that maintained the usual n-gram context but kept a beam of hidden layer configurations at each state (Auli et al., 2013). In fact, to make decoding as efficient as possible, we only keep the *single best* scoring hidden layer configuration. This approximation has been effective for lattice rescoring, since the translations represented by each state are in fact very similar: They share both the same source words as well as the same n-gram context which is likely to result in similar recurrent histories that can be safely pruned. As future cost estimate we score each phrase in isolation, resetting the hidden layer at the beginning of a phrase. While simple, we found our estimate to be more accurate than no future cost at all.

5 Experiments

Baseline. We use a phrase-based system similar to Moses (Koehn et al., 2007) based on a set of common features including maximum likelihood estimates $p_{ML}(e|f)$ and $p_{ML}(f|e)$, lexically weighted estimates $p_{LW}(e|f)$ and $p_{LW}(f|e)$, word and phrase-penalties, a hierarchical reordering model (Galley and Manning, 2008), a linear distortion feature, and a modified Kneser-Ney language model trained on the target-side of the parallel data. Log-linear weights are tuned with MERT.

Evaluation. We use training and test data from the WMT 2012 campaign and report results on French-English and German-English. Translation models are estimated on 102M words of parallel data for French-English, and 99M words for German-English; about 6.5M words for each language pair are newswire, the remainder are parliamentary proceedings. We evaluate on six newswire domain test sets from 2008 to 2013 containing between 2034 to 3003 sentences. Log-linear weights are estimated on the 2009 data set comprising 2525 sentences. We evaluate accuracy in terms of BLEU with a single reference.

Rescoring Setup. For rescoring we use either lattices or the unique 100-best output of the phrase-based decoder and re-estimate the log-linear weights by running a further iteration of MERT on the n-best list of the development set, augmented by scores corresponding to the neural network models. At test time we rescore n-best lists with the new weights.

Neural Network Training. All neural network models are trained on the news portion of the parallel data, corresponding to 136K sentences, which we found to be most useful in initial experiments. As training data we use unique 100-best lists generated by the baseline system. We use the same data both for training the phrase-based system as well as the language model but find that the resulting bias did not hurt end-to-end accuracy (Yu et al., 2013). The vocabulary consists of words that occur in at least two different sentences, which is 31K words for both language pairs. We tuned the learning rate μ of our mini-batch SGD trainer as well as the probability scaling parameter γ (3) on a held-out set and found simple settings of $\mu = 0.1$ and $\gamma = 1$ to be good choices. To prevent over-fitting, we experimented with L2 regularization, but found no accuracy improvements, probably because SGD regularizes enough. We evaluate performance on a held-out set during training and stop whenever the objective changes less than 0.0003. The hidden layer uses 100 neurons unless otherwise stated.

5.1 Decoder Integration

We compare the effect of direct decoder integration to rescoring with both lattices and n-best lists when the model is trained with a cross-entropy objective (Mikolov et al., 2010). The results (Table 1 and Table 2) show that direct integration improves accuracy across all six test sets on both language pairs. For French-English we improve over n-best rescoring by up to 1.1 BLEU and by up to 0.5 BLEU for German-English. We improve over lattice rescoring by up to 0.4 BLEU on French-English and by up to 0.3 BLEU on German-English. Compared to the baseline, we achieve improvements of up to 2.0 BLEU for French-English and up to 1.3 BLEU for German-English. The average improvement across all test sets is 1.5 BLEU for French-English and 1.0 BLEU for German-English compared to the baseline.

[7] Recombination only retains the highest scoring state if there are multiple identical states, that is, they cover the same source span, the same translation phrase and contexts.

	dev	2008	2010	syscomb2010	2011	2012	2013	AllTest
Baseline	24.11	20.73	24.68	24.59	25.62	24.85	25.54	24.53
RNN n-best rescore	24.83	21.41	25.17	25.06	26.53	25.74	26.31	25.25
RNN lattice rescore	24.91	21.73	25.56	25.43	27.04	26.43	26.75	25.72
RNN decode	25.14	22.03	25.86	25.74	27.32	26.86	27.15	26.06

Table 1: French-English accuracy of decoder integration of a recurrent neural network language model (RNN decode) compared to n-best and lattice rescoring as well as the output of a phrase-based system using an n-gram model (Baseline); Alltest is the corpus-weighted average BLEU across all test sets.

	dev	2008	2010	syscomb2010	2011	2012	2013	AllTest
Baseline	19.35	19.96	20.87	20.66	19.60	19.80	22.48	20.58
RNN n-best rescore	20.17	20.29	21.35	21.27	20.51	20.54	23.03	21.21
RNN lattice rescore	20.24	20.38	21.55	21.43	20.77	20.63	23.23	21.38
RNN decode	20.13	20.51	21.79	21.71	20.91	20.93	23.53	21.61

Table 2: German-English results of direct decoder integration (cf. Table 1).

	dev	2008	2010	syscomb2010	2011	2012	2013	AllTest
Baseline	24.11	20.73	24.68	24.59	25.62	24.85	25.54	24.53
CE RNN	24.80	21.15	25.14	25.06	26.45	25.83	26.69	25.29
+ xBLEU RNN	25.11	21.74	25.52	25.42	27.06	26.42	26.72	25.71

Table 3: French-English accuracy of a decoder integrated cross-entropy recurrent neural network model (CE RNN) and a combination with an expected BLEU trained model (xBLEU RNN). Results are not comparable to Table 1 since a smaller hidden layer was used to keep training times manageable (§5.2).

5.2 Expected BLEU Training

Training with the expected BLEU loss is computationally more expensive than with cross-entropy since each training example is an n-best list instead of a single sentence. This increases the number of words to be processed from 3.5M to 340M. To keep training times manageable, we reduce the hidden layer size to 30 neurons, thereby greatly increasing speed. Despite slower training, the actual scoring at test time of expected BLEU models is about 5 times faster than for cross-entropy models since we do not need to normalize the output layer anymore. The results (Table 3) show improvements of up to 0.6 BLEU when combining a cross-entropy model with an expected BLEU variant. Average gains across all test sets are 0.4 BLEU, demonstrating that the gains from the expected BLEU loss are additive.

6 Conclusion and Future Work

We introduce an empirically effective approximation to integrate a recurrent neural network model into first pass decoding, thereby extending previous work on decoding with feed-forward neural networks (Vaswani et al., 2013). Our best result improves the output of a phrase-based decoder by up to 2.0 BLEU on French-English translation, outperforming n-best rescoring by up to 1.1 BLEU and lattice rescoring by up to 0.4 BLEU. Directly optimizing a recurrent neural network language model towards an expected BLEU loss proves effective, improving a cross-entropy trained variant by up 0.6 BLEU. Despite higher training complexity, our expected BLEU trained model has five times faster runtime than a cross-entropy model since it does not require normalization.

In future work, we would like to scale up to larger data sets and more complex models through parallelization. We would also like to experiment with more elaborate future cost estimates, such as the average score assigned to all occurrences of a phrase in a large corpus.

7 Acknowledgments

We thank Michel Galley, Arul Menezes, Chris Quirk and Geoffrey Zweig for helpful discussions related to this work as well as the four anonymous reviewers for their comments.

References

Alexandre Allauzen, Hélène Bonneau-Maynard, Hai-Son Le, Aurélien Max, Guillaume Wisniewski, François Yvon, Gilles Adda, Josep Maria Crego, Adrien Lardilleux, Thomas Lavergne, and Artem Sokolov. 2011. LIMSI @ WMT11. In *Proc. of WMT*, pages 309–315, Edinburgh, Scotland, July. Association for Computational Linguistics.

Ebru Arisoy, Tara N. Sainath, Brian Kingsbury, and Bhuvana Ramabhadran. 2012. Deep Neural Network Language Models. In *NAACL-HLT Workshop on the Future of Language Modeling for HLT*, pages 20–28, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michael Auli and Adam Lopez. 2011. Training a Log-Linear Parser with Loss Functions via Softmax-Margin. In *Proc. of EMNLP*, pages 333–343. Association for Computational Linguistics, July.

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint Language and Translation Modeling with Recurrent Neural Networks. In *Proc. of EMNLP*, October.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:1137–1155.

Michel Galley and Christopher D. Manning. 2008. A Simple and Effective Hierarchical Phrase Reordering Model. In *Proc. of EMNLP*, pages 848–856.

Jianfeng Gao and Xiaodong He. 2013. Training MRF-Based Phrase Translation Models using Gradient Ascent. In *Proc. of NAACL-HLT*, pages 450–459. Association for Computational Linguistics, June.

Jianfeng Gao, Xiaodong He, Scott Wen tau Yih, and Li Deng. 2014. Learning Continuous Phrase Representations for Translation Modeling. In *Proc. of ACL*. Association for Computational Linguistics, June.

Joshua Goodman. 1996. Parsing Algorithms and Metrics. In *Proc. of ACL*, pages 177–183, Santa Cruz, CA, USA, June.

Xiaodong He and Li Deng. 2012. Maximum Expected BLEU Training of Phrase and Lexicon Translation Models. In *Proc. of ACL*, pages 8–14. Association for Computational Linguistics, July.

Yuening Hu, Michael Auli, Qin Gao, and Jianfeng Gao. 2014. Minimum Translation Modeling with Recurrent Neural Networks. In *Proc. of EACL*. Association for Computational Linguistics, April.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Continuous Translation Models. In *Proc. of EMNLP*, pages 1700–1709, Seattle, Washington, USA, October. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proc. of ACL Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, Jun.

Hai-Son Le, Alexandre Allauzen, and François Yvon. 2012a. Continuous Space Translation Models with Neural Networks. In *Proc. of HLT-NAACL*, pages 39–48, Montréal, Canada. Association for Computational Linguistics.

Hai-Son Le, Thomas Lavergne, Alexandre Allauzen, Marianna Apidianaki, Li Gong, Aurélien Max, Artem Sokolov, Guillaume Wisniewski, and François Yvon. 2012b. LIMSI @ WMT12. In *Proc. of WMT*, pages 330–337, Montréal, Canada, June. Association for Computational Linguistics.

Percy Liang, Alexandre Bouchard-Côté, Ben Taskar, and Dan Klein. 2006. An end-to-end discriminative approach to machine translation. In *Proc. of ACL-COLING*, pages 761–768, Jul.

Tomáš Mikolov, Karafiát Martin, Lukáš Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent Neural Network based Language Model. In *Proc. of INTERSPEECH*, pages 1045–1048.

Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černocký. 2011. Strategies for Training Large Scale Neural Network Language Models. In *Proc. of ASRU*, pages 196–201.

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

Preslav Nakov, Francisco Guzman, and Stephan Vogel. 2012. Optimizing for Sentence-Level BLEU+1 Yields Short Translations. In *Proc. of COLING*. Association for Computational Linguistics.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proc. of ACL*, pages 160–167, Sapporo, Japan, July.

Antti-Veikko I Rosti, Bing Zhang, Spyros Matsoukas, and Richard Schwartz. 2010. BBN System Description for WMT10 System Combination Task. In *Proc. of WMT*, pages 321–326. Association for Computational Linguistics, July.

Antti-Veikko I Rosti, Bing Zhang, Spyros Matsoukas, and Richard Schwartz. 2011. Expected BLEU Training for Graphs: BBN System Description for WMT11 System Combination Task. In *Proc. of WMT*, pages 159–165. Association for Computational Linguistics, July.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning Internal Representations by Error Propagation. In *Symposium on Parallel and Distributed Processing*.

Holger Schwenk, Anthony Rousseau, and Mohammed Attik. 2012. Large, Pruned or Continuous Space Language Models on a GPU for Statistical Machine Translation. In *NAACL-HLT Workshop on the Future of Language Modeling for HLT*, pages 11–19. Association for Computational Linguistics.

Martin Sundermeyer, Ilya Oparin, Jean-Luc Gauvain, Ben Freiberg, Ralf Schlüter, and Hermann Ney. 2013. Comparison of Feedforward and Recurrent Neural Network Language Models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 8430–8434, May.

Roy W. Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. 2008. Lattice Minimum Bayes-Risk Decoding for Statistical Machine Translation. In *Proc. of EMNLP*, pages 620–629. Association for Computational Linguistics, October.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with Large-scale Neural Language Models improves Translation. In *Proc. of EMNLP*. Association for Computational Linguistics, October.

Heng Yu, Liang Huang, Haitao Mi, and Kai Zhao. 2013. Max-Violation Perceptron and Forced Decoding for Scalable MT Training. In *Proc. of EMNLP*, pages 1112–1123. Association for Computational Linguistics, October.