INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

NN-grams: Unifying neural network and n-gram language models for speech recognition

Babak Damavandi, Shankar Kumar, Noam Shazeer, Antoine Bruguier

Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA
{babakd,shankarkumar,noam,tonybruguier}@google.com

Abstract

We present NN-grams, a novel, hybrid language model integrating n-grams and neural networks (NN) for speech recognition. The model takes as input both word histories as well as n-gram counts. Thus, it combines the memorization capacity and scalability of an n-gram model with the generalization ability of neural networks. We report experiments where the model is trained on 26B words. NN-grams are efficient at runtime since they do not include an output soft-max layer. The model is trained using noise contrastive estimation (NCE), an approach that transforms the estimation problem of neural networks into one of binary classification between data samples and noise samples. We present results with noise samples derived from either an n-gram distribution or from speech recognition lattices. NN-grams outperforms an n-gram model on an Italian speech recognition dictation task.

Index Terms: speech recognition, language models, neural networks

1. Introduction

A *language model* (LM) is a crucial component of natural language processing technologies such as speech recognition [1] and machine translation [2]. It helps discriminate between well-formed and ill-formed sentences in a language. Traditionally, n-gram LMs have formed the basis for most language modeling approaches. It has only been in the past few years that alternative approaches such as maximum-entropy models [3] and neural network models including feed-forward networks [4, 5, 6], recurrent neural networks (RNNs) [7] and variants such as long short term memory (LSTM) networks [8] have started outperforming n-gram models [9].

Neural network LMs have advantages over n-gram models. First, they provide better smoothing for rare and unknown words owing to their distributed word representations [9]. Neural network Models such as LSTMs have the ability to remember long-distance context, an attribute that has eluded several language models in the past. Even with these potential advantages, LSTMs and other neural network models have not been used extensively for language modeling in speech recognition because they are more resource intensive at both training and run time when compared to n-gram models. This continues to be the case despite recent efforts at speeding up training and test times using techniques such as pipelined training and variance regularization [10]. LSTMs do not scale well to the large quantities of text training data typically used for estimating n-gram LMs. This is a substantial disadvantage during training because a larger LSTM which attains a better performance than a smaller LSTM is also slower to converge. At run time, an LSTM is expensive in terms of both memory and speed relative to an n-gram model. Specifically, the output soft-max layer is computationally expensive at both training and run time if the vocabulary size is in the order of millions of words, a common characteristic of current speech recognition systems (e.g. [11]). Therefore, most neural network language modeling approaches for speech recognition have employed smaller vocabularies consisting of at most several thousands of words [5, 7]. Speech recognizers for voice search and dictation typically operate on short utterances on which n-gram models perform fairly well. This has further limited the usefulness of LSTM LMs for these tasks.

In this paper, we investigate a flavor of neural network LMs that combines the strengths of a neural network in generalizing to novel contexts with the scalability and memorization ability of an n-gram model. Our main proposal is to train a neural network that is able to learn a mapping function given both the previous history of a given word as well as the n-gram counts, which are sufficient statistics for estimating an n-gram model. By providing n-gram counts as inputs, we expect this model to learn simultaneously a function of the counts as well as the word history and estimate the probability of the current word given the history. Specifically, this model is a feed-forward neural network that takes as input the current word, K previous words, and counts for the N n-grams ending at the current word, where $N < K$. We call this model *neural network-ngrams* (*NN-grams*), to emphasize that it makes direct use of n-gram count statistics. While there have been earlier efforts at incorporating hashes of n-gram features as inputs to an RNN [7], we are not aware of a neural network model that directly takes n-gram counts as inputs.

To reduce computation, we do not include an output soft-max layer in NN-grams. While the NN-grams' score can be interpreted as a log probability of the current word given the history, the absence of a soft-max layer means that these probabilities do not necessarily sum to one over the entire vocabulary.

2. NN-grams

An LM is a probability distribution over the current word given the preceding words: $P(w_i|w_{i-1}, w_{i-2}, \ldots, w_1)$. An n-gram LM makes the assumption that the current word depends only on the previous $N-1$ words: i.e.

$$P(w_i|w_{i-1}, \ldots, w_1) = P(w_i|w_{i-1}, \ldots, w_{i-(N-1)}).$$

The architecture of the NN-grams model is given in Figure 1. The model takes as input the current word, K preceding words and counts for the N n-grams ending at the current word and estimates the log likelihood of the current word given the history:

http://dx.doi.org/10.21437/Interspeech.2016-1295

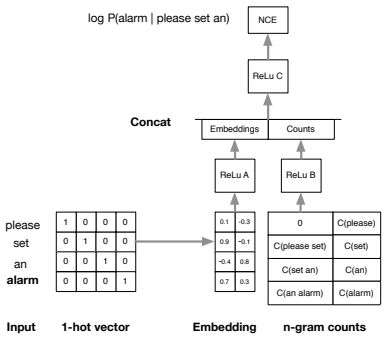

Figure 1: *Architecture of the NN-grams model. The previous words are* ***"please" "set" "an"*** *and the current word is* ***alarm****. $N = 2$, $K = 3$ and the embedding dimension is 2.*

$$f_{\text{nng}}(w_i, \ldots, w_1) = \log P(w_i | w_{i-1}, \ldots, w_{i-K}, \mathbf{c})$$

where $\mathbf{c}$ is a vector of n-gram counts of length $(K+1)N$ such that $c_1 = \text{Count}(w_i)$, $c_2 = \text{Count}(w_i, w_{i-1})$,..., $C_N = \text{Count}(w_i, w_{i-1}, \ldots, w_{i-(N-1)})$ are counts of n-grams ending at the current word, c_{N+1} through c_{2N} are counts of n-grams ending at the previous word,..., and c_{KN+1} through $c_{(K+1)N}$ are counts of n-grams ending at word w_{i-K} (See the count matrix in Figure 1 for an example), and the log probability is estimated by the neural network. Each of the words $w_i, \ldots, w_{i-K}$ is presented as a 1-hot vector to the network. The number of previous words, K, can be larger than N, the order of the n-gram counts. This enables the model to take into account longer context. Like other neural network LMs [4], the NN-grams model maps words into a high dimensional space and learns an *embedding* for each word in this space while simultaneously also learning the network parameters. The word embeddings and the n-gram counts are passed through separate layers with rectified linear unit (ReLu) activations [12] and then concatenated. The result is passed through a third ReLu layer and provided as an input to NCE. The output of the NCE layer approximates the log probability of the current word given the history and the n-gram counts. Unlike other neural network language modeling approaches [4, 5, 13], there is no explicit soft-max over the vocabulary.

2.1. Model Estimation

We train the neural network using noise contrastive estimation (NCE), a method for training unnormalized probabilistic models [14, 13, 15]. NCE transforms the estimation problem of the network into a classification problem where the goal is to differentiate between samples from the training data ($D = 1$) and those from a pre-specified noise distribution ($D = 0$). For brevity, we abbreviate the current word, w_i as w and its history $w_{i-1} \ldots, w_{i-K}, \mathbf{c}$ as h. Our goal is to fit the neural network to the training data distribution $P_{\text{data}}(w|h)$.

Suppose we have f noise samples for each training data sample, the posterior probability that the sample (h, w) arises from the training data is given by [15]:

$$P(D = 1|w, h) = \frac{P_{\text{data}}(w|h)}{P_{\text{data}}(w|h) + P_{\text{noise}}(w|h)f}.$$

We estimate this probability by replacing the data distribution

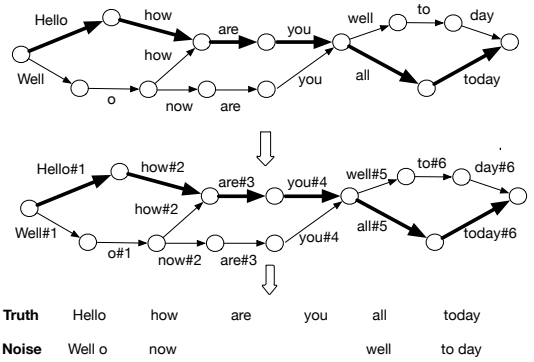

Figure 2: *Extracting speech noise samples via lattice pinching. All hypotheses in a lattice (top panel) are aligned with respect to the 1-best hypothesis (shown in bold). For each lattice edge, the alignment relative to the 1-best hypothesis is determined (middle panel). The list of noise samples is then extracted for each position (bottom panel).*

$\log P_{\text{data}}(w|h)$ with that of the neural network $\text{NN}(w, h)$:

$$\begin{aligned}\text{logit}(D = 1|w, h) &= \log\left(\frac{P(D = 1|w, h)}{1 - P(D = 1|w, h)}\right) \\ &= \log P_{\text{data}}(w|h) - \log(f) - \log P_{\text{noise}}(w|h) \\ &\approx \text{NN}(w, h) - \log(f) - \log P_{\text{noise}}(w|h).\end{aligned}$$

2.2. Noise Distributions

NCE training works best when the noise distribution is close to the data distribution. In this case, the training data samples are hard to distinguish from noise samples and the model is forced to learn about the structure of the data [14]. We experiment with two types of noise distributions. In the first type, we sample the noise word from the n-gram distribution over the words given the history. We will refer to this as *text noise*. In the second type, we sample the noise word from the word level confusions generated from a speech recognition system. We will refer to this as *speech noise*. Unlike text noise, speech noise consists of words which are acoustically confusable with the words in the training data.

Ideally, these noise samples would be words which are transcribed incorrectly by the speech recognition system when compared with a human transcription. However, the quantity of human transcriptions is limited. Therefore, we run the recognizer on utterances where human transcriptions are not available and additionally, a 1-best recognition hypothesis with high confidence exists. The noise words are the alternatives to the 1-best recognition hypothesis. We align the 1-best hypothesis to the paths in the recognition word lattice using lattice pinching [16] (Figure 2) and obtain a set of noise samples for each word in the 1-best hypothesis. Within each such set, the noise probability of a given word is its posterior probability. We exclude those words in the 1-best hypothesis which a) do not have confusions in the lattice e.g. *are* and *you* in Figure 2, and b) align to word sequences with more than more word e.g. *Hello* aligns with *well o* in Figure 2.

2.3. Count Rescaling

One of the inputs to *NN-grams* is a vector of n-gram counts. Since this count can have a large dynamic range from 0 to

several millions, we rescale the count to improve the convergence of neural network training using gradient descent [17]. If C is the original count, the rescaled count is obtained as $C' = 0.1 \log(C)$ if $C > 0$ and -1 if $C = 0$.

3. Experiments

We evaluated the *NN-gram* language model (LM) on Italian voice-search and dictation speech recognition tasks. Since the NN-gram model does not yield probability estimates that are guaranteed to be normalized, we do not report perplexities. Our test sets consisted of a voice-search (VS) set with 12,877 utterances (27.4 hours, 47,867 words) and a dictation (DTN) set with 12,625 utterances (19.2 hours, 82,121 words). All utterances were anonymized. The acoustic models were trained using convolutional, LSTM, fully connected deep neural networks as described in [18]. All LMs were trained on anonymized and aggregated search queries and dictated texts. A 5-gram LM with Katz backoff was trained using a total of 26B words, and consisted of a total of 102M n-grams. The initial word lattice was generated using this 5-gram LM and a recognition vocabulary consisting of 3.9M words.

The NN-grams model was trained on the same corpus as the 5-gram LM. Since the NN-grams model takes 6-gram counts as input, we additionally trained a 6-gram LM with Katz backoff to provide a fair baseline. Prior work [19, 20] has shown that when using pruning, n-gram models with Katz backoff outperform those with Kneser-Ney smoothing [21]. Hence, we used Katz backoff as the smoothing technique for all our n-gram language models. We limited the vocabulary size to 2M words for both models. Even though the NN-grams model has fewer parameters than the 6-gram LM (Table 1), it requires the availability of n-gram counts at run time.

LM	parameter type	# of parameters
6gram	n-grams	9.6B
NNgram	NN parameters	517M

Table 1: Model Parameters of NN-grams and 6-gram LMs.

The word lattices generated in the initial recognition pass were rescored using either the 6-gram LM or the NN-grams model. In the case of the NN-grams LM, there is no exact algorithm for rescoring the lattice. We note that there have been approximate algorithms to rescore lattices using long-span neural network language models [22, 23]. However, we did not employ these lattice rescoring methods and instead, extracted and reranked the 150-best word hypotheses from the lattice.[1] The score (log probability) of the either the 6-gram LM or the NN-grams model was interpolated with the log probability of the 5-gram LM using a fixed weight of 0.5. The 5-gram LM gave a Word Error Rate (WER) of 17.9% on VS and 11.8% on DTN.

We set the parameters K and N of the NN-grams model to 9 and 6 respectively. The model was trained until convergence with an AdaGrad optimizer [24] using a learning rate that was set to 0.01. We used a batch size of 200 in training. The dimensionality of the word embedding layer was 256. The ReLu layer that processed the embeddings (ReLu-A) had 1024 units while the Relu layer that processed the n-gram counts (ReLu-B) had 256 units. Finally, the ReLu layer that processed the concatenation of embeddings and counts (ReLu-C) had 1024 units. For NCE, we generated one noise sample for each word in the training data using text noise.

[1] If there were fewer than 150 hypotheses for an utterance, we extracted the maximum number of available hypotheses.

3.1. Comparison with n-gram LMs

We first compared the performance of NN-grams with the 6-gram LM. The results are shown in Table 2. When compared with the 6-gram LM, the NN-grams LM showed a better performance on the DTN task and an equivalent performance on the VS task. While additional gains might be potentially obtained by first rescoring the lattice with a 6-gram model followed by interpolation with NN-grams, such a system would be too slow to deploy in a speech recognition system with stringent latency requirements. Hence, we did not pursue such an interpolation.

	VS	DTN
6-gram	14.9	8.8
NN-grams	14.8	8.2

Table 2: *WER Comparison of NN-grams with 6-gram LM on voice-search and dictation.*

3.2. NN-grams components

NN-grams consist of two components: word embeddings and the n-gram counts. To determine which of these two components had a bigger impact on the overall performance of the NN-grams model, we trained the model with either one of these inputs (Figure 1). For both VS and DTN, n-gram counts were more important than word embeddings (Table 3). We expect this result considering that using n-gram counts typically improves the performance for short sentences, which is the case for both VS and DTN (Average number of words/sentence on VS and DTN is 3.7 and 6.5 respectively). For DTN, word embeddings contributed to an additional improvement in WER, that can be attributed to the longer sentence length in DTN.

NN-grams components	VS	DTN
Word-embedding,n-gram counts	14.8	8.2
Word Embedding	15.3	8.8
n-gram Counts	14.9	8.5

Table 3: *Impact of NN-grams components on WER.*

3.3. Type and Quantity of Noise Samples

We next examined whether the type and number of noise samples influenced the performance of the NN-grams model (Table 5). Since the speech noise samples can be obtained only from lattices, we restricted our training set in this experiment to only those utterances for which we were able to run the speech recognizer and generate word lattices. The training set consisted of 1.2B words from a subset of utterances derived from both voice search and dictation sources on which the 1-best recognition hypothesis had a high confidence. As a result, the WERs for these systems are worse than the system trained on 26B words (Table 2). The speech and the text noise samples were generated using the procedure described in Sec 2.2. The training data was annotated with n-gram counts derived from the 26B word corpus used in the earlier experiments.

1	Reference	Poi ha detto per le sagome dei marmi del bagno e che quando lui tornava dava le misure per fare fare i marmi
1	n-gram	poi ha detto per le sagome dei bei marmi del bagno è che quando lui torna da quale misura per fare fare i marmi
1	NN-grams	poi ha detto per le sagome dei bei marmi del bagno è che quando lui tornava dava le misure per fare fare i marmi
2	Reference	È molto più forte rispetto alle altre classi tipo Audi, Mercedes
2	n-gram	è molto pi forte rispetto a Mercedes
2	NN-grams	è molto pi forte rispetto ad altre classi tv Audi Mercedes
3	Reference	Am mo mi puoi sposare C'abbiamo la casa c'abbiamo la chiesa e c'è la sposa
3	n-gram	amò mi puoi sposare se abbiamo la casa che abbiamo la Chiesa Ecce la sposa
3	NN-grams	amò mi puoi sposare c'abbiamo la casa c'abbiamo la Chiesa e c'è la sposa

Table 4: Examples of recognition hypotheses where the NN-grams LM outperforms the n-gram LM.

# of noise samples	Text Noise		Speech Noise	
	VS	DTN	VS	DTN
1	19.2	11.6	21.7	14.5
5	19.4	12.3	20.9	14.1
10	20.1	12.6	20.4	13.9
100	17.3	10.6	17.5	12.3

Table 5: Impact of the type and number of noise samples on WER.

For each type of noise, we report WER using 1, 5, 10 and 100 noise samples. For both types of noise, the best performance was seen at 100 samples per word. The text noise outperformed the speech noise on DTN but obtained an equivalent performance on VS. It is possible that text noise, that relies on an n-gram distribution, is more suited to the dictation task where long range context is useful. In contrast, the speech noise samples which are acoustically confusable alternatives do not always have long distance dependencies. Based on these results, we could expect additional gains using 100 noise samples in the original set up with 26B words (Table 2).

3.4. Embeddings

In the NN-grams model, each word is mapped to a real valued vector. These word embeddings are key to the generalization capabilities of a neural network LM. We present examples of the top-5 nearest neighbors for two Italian words, *Roma* and *telefono* computed using the word embedding estimated in the NN-grams model (Table 6). The nearest neighbors for *Roma* are all cities in Italy while those for *telefono* consist of terms related to communication and business. In general, these neighbors in the embedding space are related to the source word, thus emphasizing the semantic nature of the space.

Roma		telefono	
Word	ED	Word	ED
Bologna	1.08	cellulare	1.33
Milano	1.09	tel	1.34
Firenze	1.15	contatti	1.41
Torino	1.16	indirizzo	1.47
Napoli	1.17	fax	1.53

Table 6: Top-5 nearest neighbors for two Italian source words: *Roma* and *telefono* computed using the NN-grams word embeddings. The Euclidean distance (ED) of each neighbor from the source word is also shown.

3.5. Examples

We present example recognition hypotheses where the NN-grams LM substantially outperformed the 6-gram LM (Table 4). In example 1, while the n-gram model prefers the common construction *torna da (come back from)*, NN-grams is either recognizing a complex construction or prefers the tense agreement between *tornava* and *dava*. In example 2, the n-gram LM prefers dropping clauses while NN-grams does not. In example 3, the NN-grams model is possibly recognizing the repeating pattern in the sentence *C'abbiamo la casa c'abbiamo la chiesa e c'è la sposa (we have a house, we have a church, we have a bride)* while the n-gram model looks independently at each of the 3 phrases, *se abbiamo*, *che abbiamo* and *Ecce*, and misrecognizes all of them. This last example may be a scenario where the long 10-word window of NN-grams gives it a distinct advantage over the n-gram LM.

4. Discussion

In this paper, we presented NN-grams, a novel neural network language modeling framework that builds upon the memorization capabilities and scalability of n-gram LMs while still allowing us to benefit from the generalization capabilities of neural networks. Our model obtains a 7% relative reduction in word error rate on an Italian dictation task. We showed that the strength of the NN-grams model comes primarily from the n-gram counts but both n-gram counts and word embeddings are important for long-form content such as dictation. We trained the model using NCE training with either text or speech noise distributions. While text noise is better for the dictation task, both noise types perform similarly for voice-search. The biggest disadvantage of the speech noise approach is that it requires decoding of utterances. Future work will investigate strategies which can directly generate acoustically confusable noise samples from only text using strategies that have been investigated in the context of discriminative language modeling [25, 26]. These strategies generate noise samples at either the phonetic or sub-phonetic (e.g. Gaussian) levels. In conclusion, the NN-grams model is a promising neural network LM that is scalable to large training texts. By avoiding the output softmax layer, it has a substantially lower overhead at training and run time compared to current neural network approaches such as LSTMs. We expect that this model will spur newer hybrid architectures which will increase the adoption of neural network approaches to language modeling.

5. Acknowledgements

We would like to thank Kaisuke Nakajima, Xuedong Zhang, Francoise Beaufays, Chris Alberti and Rafal Jozefowicz for providing crucial support at various stages of this project.

6. References

[1] F. Jelinek, *Statistical Methods for Speech Recognition*. Cambridge, MA, USA: The MIT Press, 1997.

[2] P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. . Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin, "A statistical approach to machine translation," vol. 16, no. 2, pp. 79–85, 1990.

[3] S. F. Chen, "Shrinking exponential language models," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2009, pp. 468–476.

[4] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model. 3: 1137–1155," *Journal of Machine Learning Research*, vol. 3, pp. 1137 – 1155, 2003.

[5] H. Schwenk, "Continuous space language models," *Comput. Speech Lang.*, vol. 21, no. 3, pp. 492–518, Jul. 2007.

[6] E. Arisoy, T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Deep neural network language models," 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*, ser. WLM '12. Stroudsburg, PA, USA: Association for Computational Linguistics, 2012, pp. 20–28.

[7] T. Mikolov, A. Deoras, D. Povey, L. Burget, J. Cernockỳ, and S. Khudanpur, "Strategies for training large scale neural network language models," in *ASRU*, 2011, pp. 196–201.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," *arXiv preprint arXiv:1602:02410*, 2016.

[10] X. Chen, X. Liu, M. Gales, and P. Woodland, "Improving the training and evaluation efficiency of recurrent neural network language models," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5401–5405.

[11] A. Senior, H. Sak, F. d. C. Quitry, T. N. Sainath, and K. Rao, "Acoustic modelling with CD-CTC-SMBR LSTM RNNs," in *ASRU*. IEEE, 2015.

[12] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807–814.

[13] A. Mnih and Y. W. Teh, "A fast and simple algorithm for training neural probabilistic language models," *arXiv preprint arXiv:1206.6426*, 2012.

[14] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *International Conference on Artificial Intelligence and Statistics*, 2010, pp. 297–304.

[15] A. Mnih and K. Kavukcuoglu, "Learning word embeddings efficiently with noise-contrastive estimation," in *Advances in Neural Information Processing Systems*, 2013, pp. 2265–2273.

[16] V. Goel, S. Kumar, and W. Byrne, "Segmental minimum bayes-risk decoding for automatic speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 12, no. 3, pp. 234–249, 2004.

[17] Y. LeCun, L. Bottou, G. Orr, and K. Muller, "Efficient backprop," in *Neural Networks: Tricks of the trade*, G. Orr and M. K., Eds. Springer, 1998.

[18] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4580–4584.

[19] C. Chelba, T. Brants, W. Neveitt, and P. Xu, "Study on interaction between entropy pruning and kneser-ney smoothing." in *INTERSPEECH*, 2010, pp. 2422–2425.

[20] B. Roark, C. Allauzen, and M. Riley, "Smoothed marginal distribution constraints for language modeling." in *ACL (1)*, 2013, pp. 43–52.

[21] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," in *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, vol. 1. IEEE, 1995, pp. 181–184.

[22] M. Sundermeyer, Z. Tüske, R. Schlüter, and H. Ney, "Lattice decoding and rescoring with long-span neural network language models." in *INTERSPEECH*, 2014, pp. 661–665.

[23] X. Liu, Y. Wang, X. Chen, M. J. Gales, and P. C. Woodland, "Efficient lattice rescoring using recurrent neural network language models," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 4908–4912.

[24] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *The Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[25] G. Kurata, N. Itoh, M. Nishimura, A. Sethy, and B. Ramabhadran, "Leveraging word confusion networks for named entity modeling and detection from conversational telephone speech," *Speech Commun.*, vol. 54, no. 3, pp. 491–502, Mar. 2012.

[26] P. Jyothi and E. Fosler-Lussier, "Discriminative language modeling using simulated asr errors." in *INTERSPEECH*, 2010, pp. 1049–1052.