CONVERTING NEURAL NETWORK LANGUAGE MODELS INTO BACK-OFF LANGUAGE MODELS FOR EFFICIENT DECODING IN AUTOMATIC SPEECH RECOGNITION

Ebru Arısoy, Stanley F. Chen, Bhuvana Ramabhadran, Abhinav Sethy

IBM T.J. Watson Research Center, Yorktown Heights, NY, 10598

{earisoy, stanchen, bhuvana, asethy}@us.ibm.com

ABSTRACT

Neural Network Language Models (NNLMs) have achieved very good performance in large-vocabulary continuous speech recognition (LVCSR) systems. Because decoding with NNLMs is very computationally expensive, there is interest in developing methods to approximate NNLMs with simpler language models that are suitable for fast decoding. In this work, we propose an approximate method for converting a feedforward NNLM into a back-off n-gram language model that can be used directly in existing LVCSR decoders. We convert NNLMs of increasing order to pruned back-off language models, using lower-order models to constrain the n-grams allowed in higher-order models. In experiments on Broadcast News data, we find that the resulting back-off models retain the bulk of the gain achieved by NNLMs over conventional n-gram language models, and give significant accuracy improvements as compared to existing methods for converting NNLMs to back-off models. In addition, the proposed approach can be applied to any type of non-back-off language model to enable efficient decoding.

Index Terms— Neural network language models; decoding with neural network language models.

1. INTRODUCTION

State-of-the-art decoders for automatic speech recognition generally utilize back-off n-gram language models in decoding. A back-off n-gram language model $P(w|h)$ takes the form

$$P(w|h) = \begin{cases} \hat{P}(w|h) & \text{if} \quad hw \in \mathcal{P} \\ \alpha(h)P(w|h') & \text{if} \quad hw \notin \mathcal{P} \end{cases} \tag{1}$$

where w is the current word; h is the history or last $n-1$ words; h' is the truncated history obtained by dropping the last word in h; and $\alpha(h)$ is a back-off weight that enforces normalization. The set $\mathcal{P}$ contains the n-grams hw for which we keep explicit probability estimates $\hat{P}(w|h)$; all other n-gram probablities are computed by backing off to a lower-order estimate. The distribution $P(w|h')$ as well as lower-order distributions are represented in similar fashion. Since the majority of probabilities are evaluated using back-off estimates, the model can be represented with a modest number of parameters $\hat{P}(w|h)$. Back-off language models have been extensively studied, and very efficient decoding algorithms have been developed.

Even though n-grams models are the most widely-used language models due to their simplicity and efficiency, they generalize to unseen n-grams poorly. NNLMs [1, 2, 3] were introduced to address some of the deficiencies of n-gram models, jointly learning distributed representations for words along with a probability function for word sequences. The main idea in NNLMs is to embed words into a continuous space and to perform probability estimation in this space using neural networks. Because the representations of similar unseen and seen n-grams will tend to be nearby in continuous space, NNLMs can produce reasonable probability estimates even for unseen n-grams. Feedforward NNLMs [1, 2, 3, 4] and recurrent NNLMs (RNNLMs) [5, 6] have been shown to yield both perplexity and word error rate (WER) improvements as compared to conventional n-gram language models.

Decoding directly with NNLMs is very slow, because each probability lookup is extremely expensive [7]. Consequently, NNLMs are typically applied in LVCSR within a lattice or n-best list rescoring framework. In these scenarios, the search space presented to the NNLM will be restricted by the language model used in the first-pass decoding. Integration of a NNLM into decoding avoids this restriction and may improve performance, and also avoids the increased latency introduced with rescoring.

In this paper, we propose an approximate method for converting a feedforward NNLM into a back-off language model that can be directly used in existing state-of-the-art decoders. In general, representing a NNLM exactly as a back-off language model using Eq. 1 requires storing parameters $\hat{P}(w|h)$ for all n-grams hw, seen or unseen. The main decisions to be made in approximating a NNLM as a back-off language model are how to select $\mathcal{P}$, the set of n-grams to explicitly store probabilities for; and how to select the values for $\hat{P}(w|h)$. To address these issues, we use the perspective of language model pruning, *e.g.*, entropy-based pruning [8]. Conceptually, one can imagine starting with an exact representation of a NNLM as a back-off language model, and then pruning away the n-grams resulting in the least "loss" until a model of the desired size is reached.

To calculate the loss resulting from removing an n-gram, we need to specify back-off probabilities, *e.g.*, $\hat{P}(w|h')$. It is not obvious how to extract these from an n-gram feedforward NNLM. Instead, we build m-gram NNLMs for $m = 2, \ldots, n-1$ and extract lower-order probabilities from the corresponding NNLM. In addition, it is prohibitively expensive to consider pruning all possible n-grams over a large vocabulary. Therefore, we propose a hierarchical implementation: starting from a lower-order NNLM, *i.e.*, a 2-gram model, we grow back-off models of successively higher order using higher-order NNLMs. At each level, n-gram histories are restricted to those retained in the lower-order language model, thereby making the overall pruning computation manageable. Note that this is similar to growing variable length n-gram models which are back-off language models [9, 10, 11, 12]

The rest of the paper is organized as follows. The prior work related to our paper is given in Section 2. Section 3 briefly explains NNLMs. Section 4 describes the proposed approach for converting NNLMs into back-off language models. Experiments and results are presented in Section 5. Finally Section 6 concludes the paper.

2. RELATED WORK

Several methods have been proposed for approximating NNLMs with simpler language models that can be used for efficient decoding. One approach is to use a back-off n-gram model as a variational approximation to a long-span language model such as a RNNLM [13]. To do this, text data is generated from the long-span language model and then a conventional n-gram language model is built on the simulated text data. Encouraging results were obtained with this approach even on larger data sets [14].

Another approach is to approximate a RNNLM using a weighted finite state transducer (WFST) [15]. The states of the transducer correspond to a discretization of the continuous space representation of word histories, and arc probabilities are derived from the RNNLM. WFST conversion approach outperformed the conventional bigram language model when these models were used in decoding. However this gain disappeared when the decoder output is rescored with the RNNLM. Furthermore, current state-of-the-art decoders utilize higher order n-gram models in decoding.

In [16], SuperARV language models, a syntactic language model based on Constraint Dependency Grammars, are converted to back-off n-gram models using entropy-based pruning, though few details are given on how this is done. They also propose taking an existing back-off n-gram model and replacing all $\hat{P}(w|h)$ values with probabilities from a SuperARV language model.

3. NEURAL NETWORK LANGUAGE MODELS

In neural network language models [1, 2, 3], a neural network is used to estimate language model probabilities. The input to the neural network is the words in the history and the output is the probability distribution over the predicted word. Here the basic idea is to project words into a continuous multi-dimensional feature space, and to compute the n-gram probabilities by employing the neural network. The expectation is that words that are semantically or grammatically related will be mapped to similar locations in the continuous space, allowing NNLMs to generalize well to unseen n-grams. A typical NNLM consists of input, projection, hidden and output layers. While augmenting the input layer with longer context or syntactic information [17, 18] does not increase the overall complexity much, using a large number of target words at the output layer is a challenge since the computational cost of a NNLM is mainly determined by the size of the output layer. Therefore, the output layer has targets only for the *output vocabulary* V_o, a shortlist containing the most frequent words in the vocabulary. To assign non-zero probabilities to words outside of the output vocabulary, smoothing is performed using a background language model [3]:

$$P(w|h) = \begin{cases} \beta(h) P_{\text{NNLM}}(w|h) & \text{if} \quad w \in V_o \\ P_{\text{BLM}}(w|h) & \text{if} \quad w \notin V_o \end{cases} \tag{2}$$

where $P_{\text{NNLM}}(w|h)$ and $P_{\text{BLM}}(w|h)$ represent the NNLM and background language model probabilities, respectively, and $\beta(h)$ is a history-dependent normalization constant.

4. CONVERTING A NNLM INTO A BACK-OFF LANGUAGE MODEL

For a language model to be expressed compactly as a back-off model, the probabilities of most n-grams must be proportional to their back-off probabilities (corresponding to the term $\alpha(h)P(w|h')$ in Eq. 1). This property does not hold for NNLMs, so to represent NNLM probabilities exactly over the output vocabulary requires $|V|^{n-1} \times |V_o|$ parameters in general, where V is the complete vocabulary. However, if the background language model in Eq. 2 is a back-off language model, we can take advantage of its structure to represent the overall NNLM as a back-off model. Substituting Eq. 1 for $P_{\text{BLM}}(w|h)$ in Eq. 2, we obtain

$$P(w|h) = \begin{cases} \beta(h) P_{\text{NNLM}}(w|h) & \text{if} \quad w \in V_o \\ \hat{P}_{\text{BLM}}(w|h) & \text{if} \quad w \notin V_o \bigwedge hw \in \mathcal{P}_{\text{BLM}} \\ \alpha(h) P_{\text{BLM}}(w|h') & \text{if} \quad w \notin V_o \bigwedge hw \notin \mathcal{P}_{\text{BLM}} \end{cases} \tag{3}$$

While we can represent the overall NNLM as a back-off model exactly, it is prohibitively large as noted above. The technique of *pruning* can be used to reduce the set of n-grams $\mathcal{P}$ for which we explicitly store probabilities $\hat{P}(w|h)$. In this paper, we use *entropy-based* pruning [8], the most common method for pruning back-off language models. For each n-gram, the relative entropy between the original model and the model excluding that n-gram is calculated. If the relative entropy is less than a threshold, the n-gram is pruned. To apply entropy-based pruning, we need estimates for lower-order probabilities, *e.g.*, $P(w|h')$. As it is unclear how to extract these from a feedforward NNLM, we train m-gram NNLMs for $m = 2, \ldots, n-1$ and use these to set lower-order probabilities. Unigram probabilities are set by the background language model which is trained on the language model training data.

A naive implementation for converting the overall NNLM into a back-off model is to build the entire unpruned n-gram model before performing pruning. However, this is impractical for vocabularies of any reasonable size; *e.g.*, if $|V|, |V_o| \geq 10K$ words, an unpruned 4-gram model contains at least 10^{16} n-grams.

To make pruning tractable, we propose a hierarchical algorithm where we build pruned models of increasingly higher order, and use the pruned lower-order models to constrain which n-grams are considered in the next higher-order model. In particular, given a pruned $(m-1)$-gram model, we only consider m-grams of the form hw for h that belong to the lower-order model. All other m-grams are automatically pruned, or more accurately, never added to the model in the first place. Given this restriction, we need only consider $k \times |V_o|$ m-grams for pruning at a given level if there are k items in the pruned lower-order model.

We outline the complete process for converting a 4-gram feedforward NNLM into a 4-gram back-off language model in Figure 1. First, 2-gram, 3-gram and 4-gram NNLMs and conventional language models (CLMs) are built from the training data. Our hierarchical implementation starts from 2-grams. We use Eq. 3 to combine the 2-gram NNLM with the background 2-gram CLM, giving us a 2-gram *back-off* NNLM. This model is quite large, containing $|V| \times |Vo|$ 2-grams, not including 2-grams coming from the background CLM. We apply entropy-based pruning, producing a 2-gram *pruned* back-off NNLM. The size of this model is determined by a pruning threshold. To create the 3-gram background language model, we append the 3-grams from the 3-gram CLM to the 2-gram pruned back-off NNLM and recompute the $\alpha(h)$'s to renormalize the model. Then, we repeat this procedure until the highest-order pruned back-off NNLM is obtained. When creating the initial back-off model for 3-grams and above, we add only n-grams from the NNLM that are extensions of n-grams found in the lower-order pruned back-off NNLM. Note that the proposed hierarchical approach lets us use lower-order NNLMs for backing off and same-order conventional language models for smoothing zero probability events.

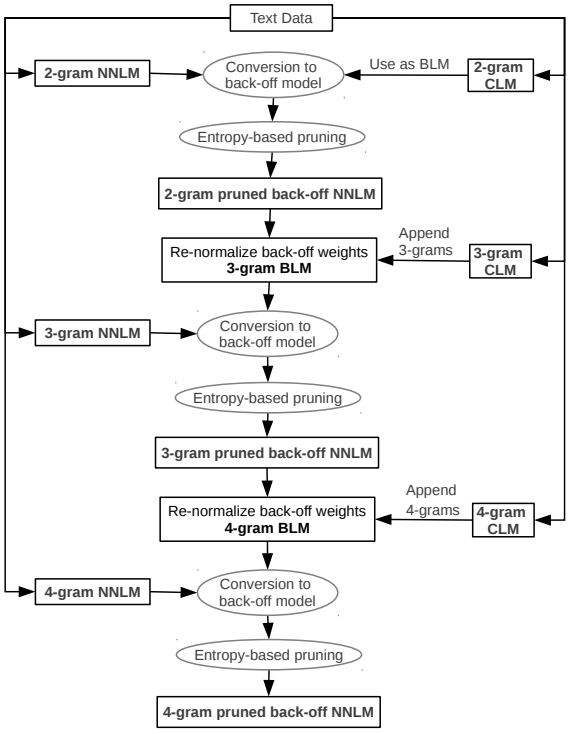

Fig. 1. Diagram showing the hierarchical implementation for converting NNLM into a 4-gram back-off language model. Conventional n-gram language model is denoted by CLM.

5. EXPERIMENTAL RESULTS

5.1. Experimental Setup

The experiments are performed on an English Broadcast News task. The baseline system is based on the 2007 IBM GALE speech transcription system [19]. The discriminatively-trained speaker-adaptive acoustic model is trained on 430h of Broadcast News audio, and we use 55M words of language model training text.[1] The baseline language model in our experiments is a 4-gram conventional language model with a 84K-word vocabulary containing a total of 46M n-grams, specifically 4M 2-grams, 15M 3-grams and 27M 4-grams. We use the *rt04* data set as the test set.

We train our NNLMs on the same 55M-word corpus. The most frequent 20K words in the vocabulary are used as the output vocabulary V_o. The 2-gram, 3-gram and 4-gram NNLMs project each input word to 120 dimensions and contain 800 hidden units. These parameters are chosen based on our previous experience with the same setup [4]. We use hierarchical NNLM training [4] with 150 classes at the output layer to speed up training. Word classes are obtained via bigram mutual information clustering [20].

Using our hierarchical pruning algorithm, we generate a 4-gram back-off language model from the NNLMs. The pruning threshold for each n-gram order is chosen to keep the number of n-grams the same as in the unpruned baseline language model. This 4-gram back-off language model is interpolated with the baseline 4-gram language model before being used in decoding. The interpolation weight is chosen to minimize the perplexity on a held-out set containing 49K words.

[1] In the original setup, a total of 350M words of training data is used.

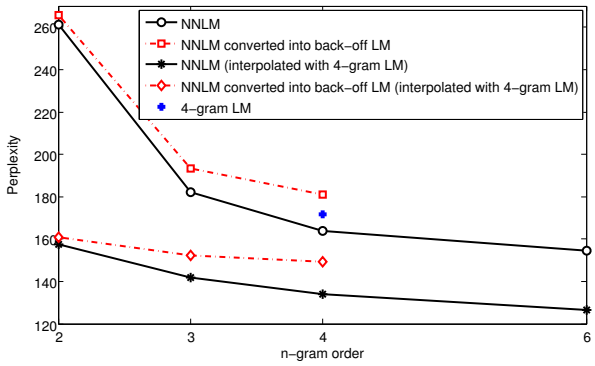

Fig. 2. Perplexity results on the held-out set.

We also train a 6-gram NNLM, projecting each input word to 120 dimensions and using 800 hidden units as before. This 6-gram NNLM is used for rescoring lattices generated by the baseline language model and by the 4-gram back-off NNLM. Before rescoring, the 6-gram NNLM is interpolated with the baseline language model and the interpolation weight is optimized on the held-out set.

5.2. Experimental Results

We first investigate the effect on perplexity of converting a NNLM into a back-off language model. Figure 2 shows the held-out set perplexity for NNLMs of various order (up to 6-gram, solid lines) as well as for the corresponding back-off language models (up to 4-gram, dashed lines). We do the conversion for only up to 4-grams since this is the highest-order model we use for decoding. In addition, we show results for before and after these models are interpolated with the baseline 4-gram language model. The "+" sign shows the perplexity for the baseline 4-gram language model. Before interpolation with the baseline, the 4-gram NNLM yields a better perplexity than the baseline, while the back-off NNLM is a little worse. After interpolation, the perplexities of both models are significantly better than the baseline, with the original NNLM achieving lower perplexities then its back-off counterpart. The perplexity differences between these models increase with increasing n-gram order. This is expected since the fraction of n-grams (out of the set of all possible n-grams) that can be retained after pruning decreases with higher n, resulting in larger approximation error.

Next, we perform decoding experiments with the 4-gram back-off NNLM interpolated with the baseline language model. Word-error rate results with this model and the baseline model are give in Table 1. The baseline WER on the rt04 test set is 14.7%. A NNLM converted into a back-off language model reduces the WER to 13.7%, yielding 1% absolute improvement over the baseline (significant at $p < 0.001$, as measured by the NIST MAPSSWE test). Then, we rescore the lattices generated by these two models with 4-gram and 6-gram NNLMs interpolated with the language models used to create the corresponding lattices. Rescoring the baseline lattices with 4-gram and 6-gram NNLMs yields WERs of 13.3% and 13.2%, respectively. Rescoring the lattices created with the back-off NNLM yields 13.0% and 12.8%, respectively. Our best result of 12.8% is a 0.4% absolute improvement (significant at $p < 0.001$) over the best result obtained by rescoring the baseline lattices. Early integration of the NNLM in decoding produces better output lattices, so that rescoring with the full NNLM yields better overall results.

Table 1. WER results with the baseline language model and with a NNLM converted into a back-off language model. Lattice rescoring results with 4-gram and 6-gram NNLMs are also reported.

Model	WER (%)
Baseline LM	14.7
+ rescore lattices with 4-gram NNLM	13.3
+ rescore lattices with 6-gram NNLM	13.2
NNLM converted into a back-off LM	13.7
+ rescore lattices with 4-gram NNLM	13.0
+ rescore lattices with 6-gram NNLM	12.8

5.3. Empirical Comparison with Other Approaches

We can relate our method to previous conversion methods by comparing how the set of retained n-grams $\mathcal{P}$ is selected, and by comparing how the parameters $\hat{P}(w|h)$ are estimated. In our approach, $\mathcal{P}$ is determined based on hierarchical entropy-based pruning and the chosen pruning threshold. The probabilities $\hat{P}(w|h)$ are taken directly from the NNLM of the appropriate order. A simple algorithm proposed in [16] is to take $\mathcal{P}$ to be the set of n-grams seen in the training data. The probabilities $\hat{P}(w|h)$ are determined in the same manner as for our algorithm. Note that this model will contain the same set of n-grams $\mathcal{P}$ as the baseline language model, but with different parameters $\hat{P}(w|h)$. To build this model, we compute the probabilities of all training data n-grams using 2-gram, 3-gram and 4-gram NNLMs. All of these n-grams and their probabilities are collected together with unigram probabilities and back-off weights are calculated. This language model is referred to as "replace probabilities" in Table 2. Using this model in decoding after interpolating with the baseline language model reduces the WER from 14.7% to 14.0% (significant at $p < 0.001$). However, our proposed approach reduces the WER to 13.7%, yielding 0.3% more gain (significant at $p = 0.021$) while selecting the same number of n-grams in $\mathcal{P}$. Thus, entropy-based pruning can select unseen n-grams that are more effective than n-grams actually occurring in the training data. Here it is important to note that in the "replace probabilities" approach, the number of n-grams remains the same, 46M, after interpolating with the baseline language model, whereas our approach results in a larger size interpolated model with 74M n-grams.

Next, we compare our approach with the variational approximation approach whereby a back-off language model is trained on text generated from a long-span language model such as a RNNLM [13]. Since NNLMs utilize a shortlist at the output layer and a background language model is used to assign probabilities to words outside of the output vocabulary V_o as given in Eq. 2, we slightly modify the implementation given in [13]. To generate words over the whole vocabulary, we need to sample text based on both the NNLM and background language model distributions. For each history, the predicted word is first sampled from the background language model distribution. If the predicted word is in the output vocabulary, this word is rejected and another word is sampled from the NNLM distribution. Otherwise, the predicted word is accepted. We generate 55M words of text, both with a 4-gram NNLM and a 6-gram NNLM. As was shown in [14], increasing the amount of the simulated text data improves the performance. However, we only simulate 55M words of text to provide a reasonable comparison for our experiments. For each simulated corpus, we train a conventional 4-gram language model and these models are used in decoding after interpolating with the baseline language model. The interpolated language models have approximately 100M n-grams. The results are given in Table 2. Language models built from the simulated text from a 4-gram NNLM and 6-gram NNLM yield word-error rates of 14.1% and 14.0%, respectively. The improvements obtained with these models over the baseline are statistically significant at $p < 0.001$. Our proposed approach yields 0.3% more gain (significant at $p = 0.014$) than the best result, 14.0%, obtained with the variational approach.

Table 2. Decoding word-error rates for various approaches for converting a NNLM to a back-off language model.

Model	WER (%)
Baseline LM	14.7
NNLM converted into a back-off LM	13.7
Replace probabilities	14.0
Simulated text from 4-gram NNLM	14.1
Simulated text from 6-gram NNLM	14.0

We can compare our approach with the variational method in terms of how n-grams parameters are selected and how parameter values are computed. The variational approach will tend to select n-grams with high marginal probabilities, though this selection will be affected by sampling noise. (If the resulting back-off model is then pruned with entropy-based pruning, the set of selected n-grams may be quite similar to those selected with our method.) For top-order n-grams, $\hat{P}(w|h)$ is taken directly from a NNLM in our method, while this parameter is set to about the same value via sampling in the variational method. For lower-order n-grams, $\hat{P}(w|h)$ is taken directly from a lower-order NNLM in our method, while these probabilities are set via Kneser-Ney smoothing, say, for the variational method. Thus, these two methods may set $\hat{P}(w|h)$ to substantially different values for lower-order n-grams.

6. CONCLUSION

In this paper, we propose a novel method for approximating a NNLM with a back-off language model to enable efficient decoding. We discuss how to represent a NNLM as a back-off language model and how to prune this model efficiently using a hierarchical algorithm. We have found that significant improvements can be obtained as compared to the baseline model through the early integration of NNLMs into decoding. Furthermore, decoding with a back-off NNLM results in better output lattices, so that lattice rescoring with a full NNLM yields additional improvements as compared to rescoring lattices generated with the baseline model. Comparisons with other approaches show that the proposed method is a promising direction for using NNLMs in decoding, though experiments on other domains and with larger data sets are needed. While we investigated only NNLMs in this work, this method can be applied to other types of language models as well, though it remains to be seen whether it will be as effective.

7. REFERENCES

[1] Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[2] Holger Schwenk and Jean-Luc Gauvain, "Training neural network language models on very large corpora," in *Proceedings of HLT-EMNLP 2005*, 2005, pp. 201–208.

[3] Holger Schwenk, "Continuous space language models," *Computer Speech and Language*, vol. 21, no. 3, pp. 492–518, July 2007.

[4] Hong-Kwang Jeff Kuo, Ebru Arisoy, Ahmad Emami, and Paul Vozila, "Large scale hierarchical neural network language models," in *Proceedings of Interspeech*, Portland, Oregon, USA, 2012.

[5] Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur, "Recurrent neural network based language model," in *Proceedings of INTERSPEECH 2010*, 2010, pp. 1045–1048.

[6] Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur, "Extensions of recurrent neural network language model," in *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*, 2011, pp. 5528–5531.

[7] Holger Schwenk and Jean-Luc Gauvain, "Connectionist language modeling for large vocabulary continuous speech recognition," in *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*, Orlando, Florida, USA, 2002, pp. 765 – 768.

[8] Andreas Stolcke, "Entropy-based pruning of backoff language models," in *Proceedings of DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne, VA, USA, 1998, pp. 270 – 274.

[9] M. Siu and M. Ostendorf, "Variable n-grams and extensions for conversational speech language modeling," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 8, no. 1, pp. 63– 75, January 2000.

[10] Vesa Siivola and Bryan Pellom, "Growing an n-gram model," in *Proceedings of Interspeech*, 2005, pp. 1309–1312.

[11] Vesa Siivola, Teemu Hirsimaki, and Sami Virpioja, "On growing and pruning kneser–ney smoothed n-gram models," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, no. 5, pp. 1617–1624, 2007.

[12] Sami Virpioja and Mikko Kurimo, "Compact n-gram models by incremental growing and clustering of histories," in *Proceedings of Interspeech - ICSLP*, Pittsburgh, PA, USA, 2006, pp. 1037–1040.

[13] Anoop Deoras, Tomas Mikolov, Stefan Kombrink, Martin Karafiat, and Sanjeev Khudanpur, "Variational approximation of long-span language models for LVCSR," in *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*, Prague, Czech Republic, 2012, pp. 5532 – 5535.

[14] Anoop Deoras, Tomas Mikolov, Stefan Kombrink, and Kenneth Church, "Approximate inference: A sampling based modeling technique to capture complex dependencies in a language model," *Speech Communication*, vol. 55, no. 1, pp. 162–177, January 2013.

[15] Gwénolé Lecorvé and Petr Motlicek, "Conversion of recurrent neural network language models to weighted finite state transducers for automatic speech recognition," in *Proceedings of Interspeech*, Portland, Oregon, USA, 2012.

[16] Wen Wang, Andreas Stolcke, and Mary P. Harper, "The use of a linguistically motivated language model in conversational speech recognition," in *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*, 2004, pp. 261–264.

[17] Ahmad Emami, *A neural syntactic language model*, Ph.D. thesis, Johns Hopkins University, Baltimore, MD, USA, 2006.

[18] Hong-Kwang Jeff Kuo, L. Mangu, A. Emami, I. Zitouni, and Y-S. Lee, "Syntactic features for Arabic speech recognition," in *Proceedings of ASRU 2009*, Merano, Italy, 2009, pp. 327 – 332.

[19] S. F. Chen, B. Kingsbury, L. Mangu, D. Povey, G. Saon, H. Soltau, and G. Zweig, "Advances in speech transcription at IBM under the DARPA EARS program," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1596 – 1608, 2006.

[20] P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer, "Class-based n-gram models of natural language," *Computational Linguistics*, vol. 18, no. 4, 1990.