Scalable Multi Corpora Neural Language Models for ASR

Anirudh Raju, Denis Filimonov, Gautam Tiwari, Guitang Lan, Ariya Rastrow

Amazon Alexa

{ranirudh,denf,tgautam,guital,arastrow}@amazon.com

Abstract

Neural language models (NLM) have been shown to outperform conventional n-gram language models by a substantial margin in Automatic Speech Recognition (ASR) and other tasks. There are, however, a number of challenges that need to be addressed for an NLM to be used in a practical large-scale ASR system. In this paper, we present solutions to some of the challenges, including training NLM from heterogenous corpora, limiting latency impact and handling personalized bias in the second-pass rescorer. Overall, we show that we can achieve a 6.2% relative WER reduction using neural LM in a second-pass n-best rescoring framework with a minimal increase in latency.

Index Terms: speech recognition, language modeling, neural language models, domain adaptation

1. Introduction

Language Models (LM) are a key component in building Automatic Speech Recognition (ASR) systems. The most common approach to building LMs for ASR systems is to learn back-off n-gram models on large text corpora. Recurrent Neural Language Models (NLM) have been shown to consistently outperform traditional n-gram language models from language modeling benchmarks [1, 2, 3, 4].

It is challenging to incorporate NLMs directly into ASR decoding partly because, unlike n-gram models, they model unlimited context history, resulting in exponential explosion in the decoder search space, and partly because in a large vocabulary ASR system the number of acoustically plausible hypotheses can be very large. There has been prior work in the area to address this issue [5] but it is still computationally very expensive. Alternatively, a more common approach is to run the ASR decoding in two passes, where the first-pass ASR system is decoded with an n-gram language model to generate a pruned search space in the form of a word lattice or an n-best hypothesis list. In the second-pass, this pruned word lattice is rescored with a stronger neural language model. One of the drawbacks of using a two-pass strategy for a real-time streaming ASR system is that the computation in the second-pass of rescoring is performed after the completion of the streaming first-pass decoding. Hence, this additional computation in the second-pass manifests itself in the form of pure latency for the ASR system. Moreover, an additional drawback is that the stronger NLM is not used during first-pass decoding, which potentially results in losing good hypotheses due to beam search.

Our main contribution in this paper is to call out and address several challenges in bringing NLMs into a latency sensitive ASR system. In particular, the challenges that we address are

- Training the NLM on multiple heterogenous corpora, effectively a domain adaptation problem
- Incorporating the NLM into the ASR system, while limiting the latency impact. This includes strategies both to reduce the second-pass NLM computation and to get benefit from the NLM in the first-pass of decoding
- Pesonalizing NLM by passing biases for classes such as contact names from the first-pass model through the NLM

The rest of the paper is organized as follows. Section 2 describes our proposed method to tackle each of challenges that we address in this work. In particular, in Section 2.1, we describe our solution to the domain adaptation problem, in Section 2.2 we describe our solution for fast inference including the usage of self-normalized models and quantization. Section 2.3 describes the usage of synthetic data generated from the NLM, to improve the first-pass model and hence the n-best hypothesis list used in second-pass rescoring.

In Section 2.4, we talk about how we handle neural rescoring when the first-pass model is a class based n-gram model [6] with personalized biases. In Section 3, we describe our experimental setup and in Section 4, we dive into our results. We conclude in Section 5 and outline directions for future work.

2. Methods and Challenges Addressed

2.1. Domain adaptation

In a practical ASR system, the LM is often trained on multiple heterogenous corpora, comprising a mix of written corpora and manually transcribed spoken text corpora from various domains. These corpora may differ in terms of their vocabulary, content, style, argots, etc [7]. We require a solution to train a neural LM on such heterogenous training data. We present two approaches below in Sections 2.1.1 and 2.1.2 to deal with this.

2.1.1. Data mixing

Training an n-gram language model on a variety of diverse corpora is straightforward. A typical strategy is to train separate n-gram models on each corpus, and combine them through linear interpolation with weights optimized to minimize perplexity on an in-domain development set [8, 9]. N-gram models have the benefit that the final linearly interpolated model, is also represented as an n-gram model. This allows for easy integration of the interpolated model into the ASR system. NLMs however, require a different approach to learning from heterogenous corpora, as a linear interpolation of NLMs, results in an ensemble model with much higher computation.

In this paper, we propose a novel solution to this problem which is simple yet effective. Parameters of neural networks are typically estimated using a variant of stochastic gradient descent, and this method relies on each minibatch being an Independent and Identically Distributed (iid) sample of the distribution we are trying to learn. Thus, we construct minibatches stochastically, by drawing samples from each corpus with probability according to its relevance weight. This has an advantage over other alternatives such as scaled loss function because it al-

lows the combination of corpora with arbitrarily different sizes and relevance weights, and in a practical system, both can vary by 2-3 orders of magnitude. For relevance weights, we construct n-gram models from each data source and optimize their linear interpolation weights on a development set. While these weights are not necessary optimal we found that they work well in practice. In the future, we plan to investigate methods for learning the relevance weights as part of NLM training procedure.

2.1.2. Transfer learning through fine tuning

One of the often used approaches to deal with the domain adaptation problem is to use fine-tuning, i.e., to train a neural network on a large out-of-domain dataset and subsequently fine-tune the parameters of the model on an in-domain dataset. Some of the parameters can optionally be fixed during the adaptation, typically those corresponding to the lower layers of the model which learn more generic transformations that are not domain specific. This has been successfully applied in computer vision [10, 11] and NLP [12]. The downside to this approach is that it does not leverage the fact that each individual out-of-domain corpus has varying relevance to the target domain. Moreover, the model also faces the challenge of catastrophic forgetting [13, 14], where the model loses past knowledge of the pre-trained weights. In order to get benefits from this method and alleviate some of its drawbacks, this approach can be combined with the data mixing strategy described in Section 2.1.1. The model is first pre-trained on the out-of-domain data, and the data mixing strategy is used during the fine tuning stage.

2.2. Fast inference solutions

2.2.1. Self-normalized models

In NLMs, the probability of word w_i given it's word history $\mathbf{h}$ is given by Eqn. 1 below:

$$p(w_i|\mathbf{h}) = \frac{exp(z_i)}{\sum_{j=1}^{|V|} exp(z_j)} = \frac{exp(z_i)}{Z(\mathbf{h})} \quad (1)$$

where z_i is the unnormalized logit corresponding to word w_i which is computed as an inner product, $z_i = exp(\mathbf{c}^T \mathbf{e}_{w_i} + b_i)$ where $\mathbf{c}^T$ is the hidden output context vector and $\mathbf{e}_{w_i}$, b_i are the output word embedding vector and bias value for word w_i. The normalization term $Z(\mathbf{h}) = \sum_{j=1}^{|V|} exp(z_j)$ is known as the partition function, and it involves a summation over all words in the vocabulary. In large vocabulary NLMs, most of the computation cost is incurred to compute the partition function that produces a proper distribution over the vocabulary as this cost is proportional to the vocabulary size.

There has been a lot of prior work on reducing this computation cost, during both training time and inference time [15, 16, 17, 18] either by approximating the partition function or by modifying the loss function to encourage the model to learn to produce approximately normalized scores (self-normalization).

The self-normalization approach allows us to compute only the scores for the query words thus eliminating the dependency on the vocabulary size. One of the approaches that can be used to train self-normalized models is to add a regularization term during training which encourages the normalization term of the softmax to be close to one [19, 20, 21]. Alternatively, Noise Contrastive Estimation (NCE) based training results in neural networks with inherent self-normalization properties [22, 23, 24, 25]. The self-normalization properties of these two broad strategies are emperically compared in [26]. While both strategies perform well computation-wise for inference, the NCE method has the benefit that it is faster during training time as well. NCE based training does not require computation of the full softmax at training time resulting in significant training speed-ups, which is independent of the output vocabulary size. In this work, we use NCE to train the Neural LMs since it has two very desirable properties of speeding up the computation during both training and inference.

2.2.2. Post-training Quantization

Quantization of the weights and activations of trained models to 16-bit fixed-point representation is performed to reduce computational cost during inference time. We perform a per-column quantization of the weight matrices, where different shifts and scales are used for each column. We found that this type of quantization performs better that using a global shift and scale for the entire matrix. This method has similarities to the per-channel quantization for convolutional networks in past work [27], which uses a different scale and shift for each convolutional kernel. While other work has explored quantization-aware training to squeeze out lower bit representations without accuracy loss, we leave this to future work.

2.3. Generating synthetic data for first-pass LM

There is a major drawback of using an NLM strictly in the second-pass to rescore lattices or n-best lists. A weaker n-gram LM is used in the first-pass and some hypotheses may be pruned, which makes them unrecoverable in the second-pass rescoring. Prior work [5] has attempted to tackle this problem by incorporating scores from the NLM into the first-pass beam search, however this is computationally expensive.

In our system, we take the approach proposed in [28], namely we construct an n-gram approximation of NLM by sampling a large text corpus from NLM and estimating an n-gram model from that corpus. Unlike [28], however, we use a *sub-word* NLM to generate synthetic data so that the generated corpus will not be limited to the vocabulary of the current version of the ASR system. Sentences containing out of vocabulary words are discarded. This way, as the vocabulary changes from version to version, we can re-use the same synthetic data.

2.4. Handling personalized first-pass LM

The first-pass LM may have classes [6] with personalized biases, for example contact names [29]. An NLM trained on general data, however, would not have good estimates for such highly personalized words or phrases. In such cases, we trust the scores of the personalized first-pass LM more than the scores of a general NLM and we do the following: Surround class content with tags `<class>` and `</class>`, and the words between the tags will retain their first-pass LM scores but they are still passed through the NLM in order to update its state so that the words after the closing tag will be estimated using the correct history.

3. Experimental Setup

In all of the experiments in this paper, we build an ASR system that targets the message dictation task. The ASR system comprises first-pass LM trained on a variety of in- and out-of-domain corpora, including written text data and transcribed speech data. The transcribed speech data is from real user-agent

interactions, and is bucketed into two separate categories - (1) message dictation specific data and (2) all other types of user-agent interactions. The transcribed messaging data which comprises of approximately 5 million words of text, is our only in-domain data corpus. The written text corpora contain over 50 billion words in total. One corpus is a 150M word long-form voicemail dataset. Although superficially similar, distributionally it is quite different from our task: for example, the average utterance length in this dataset is 67 words while our in-domain transcribed corpus has only 15 words on average.

A Kneser-Ney (KN) [30] smoothed n-gram language model is estimated from each corpus, and the final first-pass LM is a linear interpolation of these component LMs. The interpolation weights are estimated by minimizing the perplexity on target development set, in this case transcribed message dictation utterances. In experiments that use NLM generated data, we estimate a separate KN smoothed LM on the synthesized data. This n-gram LM is used as an additional component in the linear interpolation.

The NLM used in the second-pass rescoring is trained on the voicemail and message dictation corpus only, leaving out the other larger written text corpora. The reasoning behind this is that, the other corpora have a relatively low weight in the n-gram linear interpolation, and they are not that crucial to our message dictation task. The linear interpolation weights from the KN smoothed n-gram LM are 0.78 and 0.22 respectively for the message transcription and the voicemail task. In experiments which use data mixing to train the NLM, these weights are used as relavance scales for the corresponding corpora.

The NLM architecture is two LSTMP [31, 32] layers, each comprising 1024 hidden units projected down to a dimension of 512. In addition, there are residual connections [33] between the layers. The models are quantized to 16-bit fixed-point representation as described in Section 2.2.2. The NLM is used to rescore 10-best hypotheses generated from first-pass decoding.

From in-domain corpus, we extract the vocabulary of 60k most frequent words. All NLM models use this vocabulary and out of vocabulary tokens are mapped to `<unk>`. Note that the first-pass ASR system has a larger vocabulary, 160k, plus new words can be introduced via personalized classes. In rescoring experiments (but not for perplexity computation), we scale the probability of `<unk>` token by a factor of 10^{-5}, i.e., we assume a uniform distribution over the "missing" vocabulary.

4. Results and Discussion

4.0.1. Domain adaptation experiments

Table 1 shows perplexity results comparing NLMs trained on a single data source against different domain adaptation methods described in Sections 2.1.1 and 2.1.2: mixing multiple corpora, applying transfer learning (fine-tuning), and combining both methods. First, we confirm that our voicemail corpus is indeed out-of-domain: perplexity of an NLM model trained on just that data is 116.0, more than double the perplexity of a model trained on in-domain corpus only, 55.8. Next, we study the impact of the two domain adaptation strategies. The results from Table 1 show a 12.6% relative improvement in perplexity using transfer learning compared to a baseline trained on in-domain message dictation data only. The model is trained through fine-tuning i.e. initially learning a model on the out-of-domain voicemail corpus and further fine-tuning on in-domain message dictation data. This is in line with prior work in literature. More interestingly, by training the model directly on data mixed from both messaging and voicemail data with the relevance weights estimated from an interpolated KN smoothed n-gram model, we can obtain a 13.4% improvement in perplexity compared to the baseline model. These results are very promising, since the models trained with the data mixing approach, provide slightly better perplexity results, and train significantly faster than the transfer learning approach. The disparity in training speeds is because the transfer learning approach requires two rounds of training, with the pre-training round performed on a significantly larger out-of-domain corpus (which is usually the case, since there is far lesser in-domain training data available). In the data-mixing approach, the model converges much quicker, seeing several epochs of the in-domain data and fewer epochs (possibly lesser than one) on the out-of-domain data. This is for the simple reason that the in-domain corpus is much smaller and the sampling weights of the two corpora is typically skewed towards the in-domain corpus (0.78 in our experiments).

Finally, it is possible to combine the two approaches described above, i.e., first pre-training the model on out-of-domain data and then fine-tuning on a mixture of in-domain and out-of-domain data. This results in a 16.1% relative improvement in perplexity compared to the baseline, which is better than each of the individual approaches alone. In all future experiments, we use the best NLM obtained from both data-mixing and fine-tuning

Table 1: *Perplexity results for domain adaptation. Voicemail corpus is out-of-domain for the message dictation task. "Mix" refers to the data mixing approach*

Pretrain Corpus	Train Corpus	PPL
-	Voicemail	116.0
-	Messaging	55.8
Voicemail	Messaging	48.8
-	Voicemail + Messaging mix	48.3
Voicemail	Voicemail + Messaging mix	46.8

4.0.2. Inference speed impact of self-normalized LM

Table 2 shows that the perplexity of unnormalized and normalized models are very close, which will allow us to use unnormalized probabilities for the second-pass rescoring saving a bulk of the inference computation time. In order to show this, we compare the p50 and p90 percentiles for latency added purely due to the second-pass rescoring. This is shown in Table 3, where the rescoring latency of the self-normalized NCE LM is lower than the softmax LM by about 700ms at p50 and 3100ms at p90 percentiles.

Table 2: *Perplexity results comparing normalized and unnormalized NCE models on a voicemail development set. Unnormalized probabilities do not include the softmax normalization factor*

Model	PPL
Softmax NLM	19.42
NCE NLM (normalized)	19.95
NCE NLM (unnormalized)	20.44

4.0.3. WER Impact from NLM

Table 3 shows that we are able to obtain 1.6% relative WER reduction from using NLM-generated synthetic data. Since this is just an update to the first-pass LM there is no increase in latency. These results are in line with the perplexity improvements seen in Table 4 with the inclusion of NLM-generated synthetic data. Note that the NLM used to generate the synthetic data is a subword LM, discarding sentences with out of vocabulary words with respect to the first-pass ASR system. In Table 4, the perplexity number of the NLM reported in the last row is of a softmax word-based LM, included as a fair reference for comparison with the KN smoothed n-gram perplexity numbers. Finally, performing a 10-best second-pass rescoring using self-normalized NLM gets us a net relative WER reduction of 6.2%. Note that the WER reduction from both, the softmax NLM and self-normalized NCE NLM, are very similar and in line with the perplexity numbers of Table 2.

Table 3: *Relative Word Error Rate Reduction (WERR) and rescoring latency numbers, showing the effect of including NLM synthetic data and rescoring with softmax vs. unnormalized NCE NLM*

LM		WERR	Rescoring latency	
First-pass	**Second-pass**		P50	P90
Baseline	-	-	-	-
+Syn data	-	1.6%	-	-
+Syn data	Softmax NLM	6.3%	767ms	3396ms
+Syn data	NCE NLM	6.2%	65ms	285ms

Table 4: *Perplexity results comparing n-gram LMs with and without NLM generated synthetic data on a message dictation test set. "Msg" refers to transcribed message dictation data, "Synthetic Msg" refers to data generated from NLM and "Others" refers to all other available corpora*

LM	Train data	PPL
KN-4g	Msg	63.71
KN-4g-Interp	Msg + Others	60.81
KN-4g-Syn	Synthetic Msg	58.34
KN-4g-Interp-Syn	Msg + Synthetic Msg + Others	58.11
NLM	Msg	46.85

4.1. Impact from personalized bias from first-pass LM

Recognition of contact names is important for a message dictation application. The ASR system in this paper is specifically focused on message dictation payload, where the recognition of contact names, within the message are important from a user experience perspective. For example, a fairly common message such as "hey john how was your day" requires accurate recognition of the name "john", which is challenging for a rare or out of vocabulary name. This benefits from the usage of a class-based LM, with a single class for contact names, since it allows us to use personalized contact names list for biasing the model towards user specific information. This is measured through the Entity WER metric. To measure this, we tag each word in our test data using an in-house Named Entity Recognition (NER) tagger. The Entity WER is defined as $(num\_substitutions + num\_deletions)/num\_reference\_words$. The hypothesis and reference are aligned in order to calculate the number of substitutions and deletions corresponding to the tagged reference words. Note that we do not include insertions due to difficulty in attributing whether an insertion error was caused by the entity or the other surrounding words. The results in Table 5 showing the Entity WERR for Person names, demonstrate that by appropriately handling the contacts class in the NLM through class tags, we are able do slightly better than a naive approach of rescoring these with the NLM. These class tags enabled us to induce personalized bias in the rescorer by retaining the first-pass scores for the contact names, ignoring the score from the NLM but using the word input to update the LSTM state information. This method was previously described in detail in Section 2.4. Overall, this enabled accuracy improvements for contact name recognition, which is important to user experience.

Table 5: *Relative Entity Word Error Rate Reduction (WERR) of contact names, with and without personal bias in rescorer*

LM		Entity
First-pass	**Second-pass**	**WERR(%)**
KN-4g-Interp-Syn	-	-
KN-4g-Interp-Syn	NLM	9.18%
KN-4g-Interp-Syn	NLM + bias	9.56%

5. Conclusions and Future Work

In this work, we addressed several challenges for an NLM to be used in a practical large-scale ASR system. In particular, training an NLM from multiple heterogenous corpora using a novel data mixing strategy, along with transfer learning based on fine-tuning that provided 16.1% relative improvement in perplexity compared to a baseline trained on in-domain data only. Subsequently, we presented work to limit latency impact of the models. The usage of self-normalized LM helped us to reduce the added latency by 700ms and 3100ms at the 50th and 90th percentiles, compared to using softmax based LMs. We were able to obtain a 1.6% relative WERR by generating synthetic data from the NLM and incorporating that into an n-gram model used in the first-pass beam search decoding. Overall, this provided a net WERR of 6.2% relative along with 10-best rescoring. Finally, we showed that we were able to get accuracy improvements for contact names, using personalized list information, by using classes in the first-pass LM and appropriately handling them in the NLM rescoring through class tags. In the future, we plan to evaluate the data mixing strategy in handling more than two corpora as well as investigating methods for optimizing data mixing weights as part of NLM training procedure.

6. References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, no. 6, pp. 1137–1155, 2003.

[2] H. Schwenk, "Continuous space language models," *Computer Speech & Language*, vol. 21, no. 3, pp. 492–518, 2007.

[3] T. Mikolov, M. Karafit, L. Burget, J. . ernock, and S. Khudanpur, "Recurrent neural network based language model," in *INTERSPEECH*, 2010, pp. 1045–1048.

[4] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," *arXiv preprint arXiv:1602.02410*, 2016.

[5] T. Hori, Y. Kubo, and A. Nakamura, "Real-time one-pass decoding with recurrent neural network language model for speech recognition," in *ICASSP 2014 - 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 6364–6368.

[6] P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, "Class-based n-gram models of natural language," *Computational linguistics*, vol. 18, no. 4, pp. 467–479, 1992.

[7] J. R. Bellegarda, "Statistical language model adaptation: review and perspectives," *Speech communication*, vol. 42, no. 1, pp. 93–108, 2004.

[8] A. Gandhe, A. Rastrow, and B. Hoffmeister, "Scalable language model adaptation for spoken dialogue systems," in *2018 IEEE Spoken Language Technology Workshop (SLT)*, 2018, pp. 907–912.

[9] A. Raju, B. Hedayatnia, L. Liu, A. Gandhe, C. Khatri, A. Metallinou, A. Venkatesh, and A. Rastrow, "Contextual language model adaptation for conversational agents." in *Interspeech 2018*, 2018, pp. 3333–3337.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[11] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[12] L. Mou, Z. Meng, R. Yan, G. Li, Y. Xu, L. Zhang, and Z. Jin, "How transferable are neural networks in nlp applications?" *arXiv preprint arXiv:1603.06111*, 2016.

[13] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psychology of Learning and Motivation*, vol. 24, pp. 109–165, 1989.

[14] R. Ratcliff, "Connectionist models of recognition memory: constraints imposed by learning and forgetting functions," *Psychological Review*, vol. 97, no. 2, pp. 285–308, 1990.

[15] T. Mikolov, S. Kombrink, L. Burget, J. Cernocky, and S. Khudanpur, "Extensions of recurrent neural network language model," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011, pp. 5528–5531.

[16] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model." in *AISTATS*, 2005.

[17] X. Chen, X. Liu, Y. Wang, M. J. F. Gales, and P. C. Woodland, "Efficient training and evaluation of recurrent neural network language models for automatic speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 11, pp. 2146–2157, 2016.

[18] Y. Shi, W.-Q. Zhang, M. Cai, and J. Liu, "Variance regularization of rnnlm for speech recognition," in *ICASSP 2014 - 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 4893–4897.

[19] J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. M. Schwartz, and J. Makhoul, "Fast and robust neural network joint models for statistical machine translation," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014, pp. 1370–1380.

[20] W. Chen, D. Grangier, and M. Auli, "Strategies for training large vocabulary neural language models," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2016, pp. 1975–1985.

[21] J. Andreas and D. Klein, "When and why are log-linear models self-normalizing?" in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015, pp. 244–249.

[22] A. Mnih and Y. W. Teh, "A fast and simple algorithm for training neural probabilistic language models," *international conference on machine learning*, pp. 419–426, 2012.

[23] A. Vaswani, Y. Zhao, V. Fossum, and D. Chiang, "Decoding with large-scale neural language models improves translation," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 1387–1392.

[24] X. Chen, X. Liu, M. J. F. Gales, and P. C. Woodland, "Recurrent neural network language model training with noise contrastive estimation for speech recognition," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 5411–5415.

[25] B. Zoph, A. Vaswani, J. May, and K. Knight, "Simple, fast noise-contrastive estimation for large rnn vocabularies." in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 1217–1222.

[26] J. Goldberger and O. Melamud, "Self-normalization properties of language modeling," *international conference on computational linguistics*, pp. 764–773, 2018.

[27] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper." *arXiv preprint arXiv:1806.08342*, 2018.

[28] A. Deoras, T. Mikolov, S. Kombrink, M. Karafit, and S. Khudanpur, "Variational approximation of long-span language models for lvcsr," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011, pp. 5532–5535.

[29] P. Aleksic, C. Allauzen, D. Elson, A. Kracun, D. M. Casado, and P. J. Moreno, "Improved recognition of contact names in voice commands," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5172–5175.

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

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

[32] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *arXiv preprint arXiv:1402.1128*, 2014.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.