INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

Neural Language Modeling With Implicit Cache Pointers

Ke Li[1], Daniel Povey[2], Sanjeev Khudanpur[1,3]

[1]Center for Language and Speech Processing, Johns Hopkins University, USA
[2]Xiaomi Corporation, Beijing, China
[3]Human Language Technology Center of Excellence, Johns Hopkins University, USA

{kli26,khudanpur}@jhu.edu, dpovey@gmail.com

Abstract

A cache-inspired approach is proposed for neural language models (LMs) to improve long-range dependency and better predict rare words from long contexts. This approach is a simpler alternative to attention-based pointer mechanism that enables neural LMs to reproduce words from recent history. Without using attention and mixture structure, the method only involves appending extra tokens that represent words in history to the output layer of a neural LM and modifying training supervisions accordingly. A memory-augmentation unit is introduced to learn words that are particularly likely to repeat. We experiment with both recurrent neural network- and Transformer-based LMs. Perplexity evaluation on Penn Treebank and WikiText-2 shows the proposed model outperforms both LSTM and LSTM with attention-based pointer mechanism and is more effective on rare words. N-best rescoring experiments on Switchboard indicate that it benefits both very rare and frequent words. However, it is challenging for the proposed model as well as two other models with attention-based pointer mechanism to obtain good overall WER reductions.

Index Terms: RNNLM, Transformer, cache model, pointer component, automatic speech recognition

1. Introduction

Neural language models (LMs) are an important module in automatic speech recognition (ASR) [1, 2, 3]. Standard recurrent neural network language models (RNNLMs) make predictions based on a fix-sized hidden vector, making modeling long-range dependency challenging. Although LSTMs outperform vanilla RNNs, it has been observed that they usually retain only a relatively short span of context [4, 5]. Memory augmented models and attention mechanism have been proposed to increase the hidden state's capacity to retrieve information from hidden states in the more distant past. Though improved performance has been reported, RNNLMs with the standard softmax output still struggle with rare or unknown words, even with attention.

Since the self-attention architecture was proposed [6], deep Transformers have demonstrated state-of-the-art performance on natural language processing tasks [7, 8, 9]. Transformer-based LMs have outperformed RNNLMs on large corpora and been used in rescoring stage in ASR systems [10, 11]. However, their ability to capture long-term dependency, e.g. self-trigger effects (word repetitions), remains unclear.

In real scenarios, especially in conversations, after a word or phrase is spoken, it is highly likely to be spoken again [12, 13]. These self-triggers or topic-word effects can be captured by cache models, which stores the unigram distribution of recently seen words. Cache models adapt pre-trained LMs to local contexts (decoded hypotheses) in ASR systems and hence can improve ASR performance [14, 15]. Usually, cache models are integrated in pre-trained models at test time. It is a lightweight approach as no model retraining is required, while it may not be optimal. Effectively incorporating them in training stages and enabling neural LMs to learn to adapt to recent history remain to be explored.

In this work, we propose a cache-inspired approach for neural LMs to improve the capability of modeling long-term dependency, especially for rare words. The output is extended by a predefined size L to represent L preceding words in history. The pre-softmax activation of the L units, like the other pre-softmax units, is computed by a linear transformation of the hidden state or context vector, and then appended to the output before the softmax layer, as shown in Figure 1. The training loss is still cross entropy. However, unlike standard training, wherein supervision comes from a vocabulary-sized one-hot vector encoding the predicted word, the supervision vector is now L bits longer and contains additional ones in each history position where the word is the same as the predicted one.

The extended output and modified supervision implicitly enable learning *from where* in history *to copy*. While it may still be difficult for the model to learn which words are particularly likely to be self-triggers, i.e. *when to copy*. To provide a mechanism for this, one additional unit is introduced in the pre-softmax layer (but not included in the softmax computation) to capture the probability that the current word may be a self-trigger. At each word position, activations from these additional units in the L previous positions are added to the L extended output units.

Though cache-inspired neural LMs for improving long-range dependency have been proposed and demonstrated superior performance than LSTMs in terms of perplexity, to our best knowledge, their effect on ASR accuracy remains to be explored. In this study, we evaluate neural LMs on ASR tasks. We also apply the proposed approach to Transformer architecture to verify if cache-based information is still beneficial.

2. Related Work

In this section, we briefly introduce related work about approaches to improve performance on rare words and long-term dependency for sequence modeling problems including neural language modeling and machine translation [16, 17, 18, 19, 20, 21]. Vinyals et al. [16] introduces an attention-based pointer network to select items from the input as output. It has been shown to help on geometric problems [16]. The pointer network can also improve performance of text summarization [17, 18] and alleviate issues of rare or unknown words in neural machine translation [18].

This work was partially supported by unrestricted gifts from Facebook and Applications Technology (AppTek).

http://dx.doi.org/10.21437/Interspeech.2020-3020

For neural LMs, similar ideas have been proposed to better model long-range dependency [19, 20]. The most relevant work is the pointer sentinel mixture model (PSMM), a mixture model of a standard LSTM and an auxiliary pointer network which captures the unigram distribution of history words via attention [19]. The mixture weight is jointly optimized. A similar mixture model, neural cache model [20], differs from PSMM in aspects such as the query vector for computing attention scores is hidden state itself instead of a projected version and it does not require model retraining. The motivation of dynamic evaluation [21] is similar to the neural cache model, but the implementation is different: it adjusts model parameters via gradient updates based on partial predicted sequences during test time. It may be viewed as a modified version of the dynamic updating method proposed by Mikolov et al. [1].

3. Proposed Model

Language modeling can be framed as predicting the next word (target) given preceding words (history). It usually can be observed that some words tend to be much more likely targets once they have occurred in the history. PSMM learns to "reproduce" a word from recent history by an attention-based pointer network. And its mixture weight is computed by a specially designed gating mechanism. Though the PSMM achieves lower perplexity than standard LSTM, the attention and gating mechanisms are relatively complex and not the only approach to do so. We aim to achieve a similar effect with simpler models.

3.1. Pointer Component

Let us denote the hidden state of an RNNLM at time step t as $\mathbf{h}_t$. Conventionally, the RNNLM output $\mathbf{y}_t$ is determined as

$$\mathbf{y}_t = \text{softmax}(\mathbf{W}\mathbf{h}_t + \mathbf{b}), \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{V \times H}$, $\mathbf{b} \in \mathbb{R}^V$, and $\mathbf{y}_t \in \mathbb{R}^V$, with V and H being the vocabulary size and hidden state dimension, respectively.

We extend the output dimension by a predefined size L. The extended part represents the L immediately preceding words in history. Activation of these L extended units, denoted as $\mathbf{p}_t$ in Figure 1, is computed via a linear projection of $\mathbf{h}_t$ from the last hidden layer of the RNNLM. We thus have

$$\mathbf{p}_t = \mathbf{W}_p \mathbf{h}_t, \tag{2}$$

$$\mathbf{z}_t = \text{concat}(\mathbf{W}\mathbf{h}_t + \mathbf{b}, \mathbf{p}_t), \tag{3}$$

$$\mathbf{y}_t = \text{softmax}\left(\mathbf{z}_t\right), \tag{4}$$

where $\mathbf{W}_p \in \mathbb{R}^{L \times H}$, $\mathbf{z}_t \in \mathbb{R}^{V+L}$, and $\mathbf{p}_t \in \mathbb{R}^L$. Applying softmax on $\mathbf{z}_t$ generates the extended output $\mathbf{y}_t \in \mathbb{R}^{V+L}$.

Since the L extended outputs indicate *where* to copy *from* the history, we call $\mathbf{p}_t$ the *pointer* component of our model. It only introduces $L \times H$ additional parameters.

The objective for training a neural LM is to maximize the log likelihood of training data. The loss function is written as

$$L(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \log(\mathbf{y}_t \cdot \mathbf{s}_t), \tag{5}$$

where T is the total number of words in the training data, $\mathbf{s}_t$ is the supervision vector, and $\cdot$ is vector dot product operation. In conventional training, $\mathbf{s}_t$ is a one-hot vector with 1 in the index of the target word. To train the proposed neural LM with the pointer component, the supervision vector $\mathbf{s}_t$ is set to have

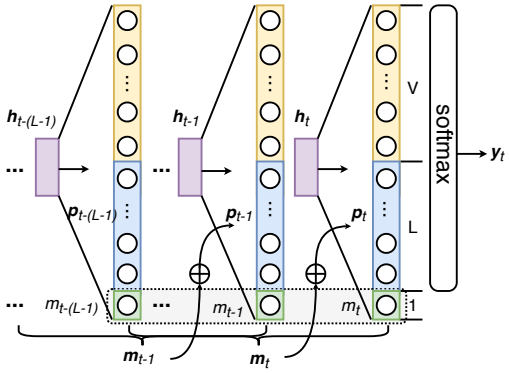

Figure 1: *Neural LMs with implicit cache pointers.*

additional ones in history positions where the the target was previously seen. So $\mathbf{s}_t$ is an *at-least-one-hot* vector.

3.2. Memory Augmented Pointer Component

The pointer component and the modified training supervision makes an RNNLM be aware of where to copy from the history. However, it may still be challenging for the model to memorize which words are particularly likely to reoccur, i.e. are "bursty". To learn the burstiness of words, one additional unit, denoted by m_t is introduced alongside the pointer $\mathbf{p}_t$, as shown in Figure 1.

This additional unit is computed by a vector dot product of the hidden state $\mathbf{h}_t$ and a parameter vector with the same dimension as $\mathbf{h}_t$, but is not used in computing the softmax. It influences the probability that a word may repeat through $\mathbf{p}_t$. Specifically, these additional units from the L immediately preceding word positions are concatenated to form

$$\mathbf{m}_t = \text{concat}(m_{t-(L-1)}, ..., m_t), \tag{6}$$

where $\mathbf{m}_t \in \mathbb{R}^L$, and $\mathbf{m}_t$ is element-wisely added to the pointer component $\mathbf{p}_t$, i.e.

$$\mathbf{p}_t := \mathbf{p}_t + \mathbf{m}_t. \tag{7}$$

Thus m_t influences the output $\mathbf{y}_t$ indirectly by modifying $\mathbf{p}_t$ in (3), which in turn is a part of $\mathbf{z}_t$ in (4).

Compared with an RNNLM, the memory augmented pointer component only has $(L+1) \times H$ total additional parameters while a PSMM has extra $H^2 + 2H$ parameters. Without attention and gating mechanism, the proposed model is simpler than PSMM and has fewer extra parameters when $L \leq H$.

This pointer mechanism described above is for RNNLMs, while it can also be easily incorporated into Transformer-based LMs. In the latter, the context vector from the last Transformer block is treated as the hidden state in RNNLMs.

4. Experimental Setup

We conduct experiments on two text datasets, Penn Treebank (PTB) and WikiText-2 [19], and two ASR corpora, Switchboard (SWBD) and Wall Street Journal (WSJ). We use Kaldi RNNLM [3] for data preprocessing on SWBD, e.g. including the English Fisher corpus, and WSJ. Sentences in SWBD+Fisher interleave conversation turns, as derived from time-information in transcriptions. Statistics of the datasets are shown in Table 1 ("sent len" is average sentence length).

We develop baselines with both LSTM and Transformer-based LMs. Model details are present in Table 2. Plain LSTMs

Table 1: *Statistics of datasets used in experiments.*

Dataset	# words*	Vocab	sent len	OOV (train / dev / test)	Style
PTB	929K	10K	21	4.8% / 4.7% / 5.8%	written
WikiText-2	2M	33K	22	2.6% / 5.4% / 6.2%	written
SWBD+Fisher	34M	30K	10	8.9% / 0.0% / 5.8%	spoken
WSJ	39M	123K	23	4.6% / 4.6% / 5.6%	written

* The end of sentence token is included in the count of training words.

are baselines from each dataset except for WSJ. We has a stronger baseline for PTB and WikiText-2: AWD-LSTM [22] with frequency-agnostic word embeddings [23], denoted by Frage-AWD-LSTM. For SWBD+Fisher, the stronger baseline is a Transformer LM with self-attention [6]. We only experiment with Transformer architecture on WSJ. Given our academic computational resources, we were unable to make comparisons with even stronger baselines, e.g. GPT and BERT, or with the optimized architectures of [24], which requires industrial-strength resources.

All neural LMs are on word level, implemented with Pytorch, and optimized via SGD[1]. We tie the embedding and output matrices in all setups. The dropout rate for PTB and WikiText-2 is 0.5, while for SWBD+Fisher it is 0.1 for both LSTM and Transformer LMs. Parameters of Frage-AWD-LSTMs not listed in Table 2 follow the settings in [23].

Table 2: *Details of neural network dimensions for various LMs.*

Model	Corpus	Layers	Units	Heads
Plain LSTM	All (except for WSJ)	2	650	-
Frage-AWD-LSTM	PTB/WikiText-2	3	1150	-
Transformer	SWBD+Fisher/WSJ	6	512/768	8

For ASR experiments on SWBD and WSJ, we use the Kaldi toolkit [25] to train acoustic models and perform N-best rescoring. Acoustic models are factorized TDNNs [26], trained using the LF-MMI objective [27]. We do not include Fisher audio to train acoustic models for SWBD. To rescore each of the N hypotheses for an utterance, we find it useful to initialize the initial LM state with the last LM state of the best hypothesis for the previous utterance.

5. Experiments

5.1. Perplexities on PTB and WikiText-2

We first compare the proposed model with PSMM and neural cache under the plain LSTM setup. Perplexities on PTB and WikiText-2 are shown in Table 3. The performance gap between the PSMM in the paper [19] and ours is mainly caused by different implementations of truncated back-propagation through time (BPTT). They use an explicit truncated BPTT while we follow the normal way discussed in [19] considering efficiency and convenience of data preprocessing. We first concatenate all text words and then chunk them with fixed size L. So, if the truncated BPTT length is L, each training word on average experiences $L/2$ instead of L time-steps for back-propagation. This means each training word sees $L/2$ history words on average.

[1]For the Transformer LMs, we tried Adam with the learning rate schedule proposed in [6], but failed to get better performance than SGD.

Table 3: *Perplexities on PTB and WikiText-2 (plain LSTMs).*

Model	PTB			WikiText-2		
	#Params	Dev	Test	#Params	Dev	Test
5gram KN [28]	2M	-	141.2	-	-	-
LSTM (medium) [29]	20M	86.2	82.7	-	-	-
PSMM [19]	21M	72.4	70.9	47M[2]	84.8	80.8
LSTM	13.3M	73.6	71.9	28.5M	89.1	84.8
PSMM (Ours)	13.7M	73.5	71.6	28.9M	86.8	82.8
LSTM + Neural Cache (L=50)	13.7M	69.3	68.5	28.9M	81.3	77.0
Proposed w/o Memory Aug	13.3M	70.1	69.8	28.5M	80.6	76.7
Proposed w/ Memory Aug	13.3M	**68.1**	**67.8**	28.5M	**78.2**	**74.3**

In Table 3 "Memory Aug" refers to the memory augmented pointer. We set history length as 100, equal to the truncated BPTT length. Setting $L = 50$ for neural cache is a fair comparison with others. Results on both datasets show that memory augmentation provides further improvement on top of the pointer component. And with memory augmentation, the proposed model outperforms the rest on both datasets. In subsequent tables, "Proposed" refers to LMs with the memory augmented pointer.

To verify whether the proposed approach is robust, we conduct experiments on a stronger baseline Frage-AWD-LSTM setup [23]. We reproduced their results and implemented the proposed approach on top of theirs, without tuning meta-parameters. Perplexity results in Table 4 shows that the proposed model on both datasets achieves better results than Frage-AWD-LSTM. Further improvements are observed with increasing the history length from 50 to 100, as expected. We also observe complementary effects of the proposed model and neural cache model.

Table 4: *Perplexities on PTB and WikiText-2 (Frage-AWD-LSTM setup).*

Model	PTB			WikiText-2		
	#Params	Dev	Test	#Params	Dev	Test
Frage-AWD-LSTM [23]	24M	58.1	56.1	33M	66.5	63.4
Frage-AWD-LSTM (Ours)	24M	57.7	55.3	33M	64.6	62.1
Proposed (L = 50)	24M	55.2	53.9	33M	60.7	58.5
Proposed (L = 100)	24M	54.2	53.5	33M	60.2	57.5
Proposed + Neural Cache(L=100)	24M	**53.0**	**52.5**	33M	**58.0**	**55.6**

5.2. Perplexities on SWBD and WSJ

We experiment with both LSTM- and Transformer-based LMs on SWBD. Perplexity on dev set from Kaldi RNNLM is 50. The history length as well as BPTT length is set to 100 for the proposed model and PSMM. Results of Pytorch trained models are in Table 5. For LSTM-based models, the proposed ap-

Table 5: *Perplexities on SWBD.*

Model	#Params	Dev	Eval'00
LSTM	26.5M	47.0	41.7
PSMM	26.9M	45.6	39.5
LSTM + Neural Cache (L=50)	26.5M	45.1	39.6
LSTM + Neural Cache (L=100)	26.5M	**44.6**	**39.1**
LSTM + Proposed	26.5M	45.9	40.4
Transformer w/o positional embedding [6]	25.0M	51.6	44.4
Transformer with positional embedding	25.1M	46.8	41.5
Transformer + Proposed	25.1M	45.0	40.2

proach outperforms the baseline LSTM, but performs slightly worse than the PSMM and neural cache models. For Transformer LMs, the proposed approach also achieves better perplexity than the two Transformer baselines.

We notice the performance gains of the proposed model over both LSTM and Transformer baselines on SWBD are smaller than on PTB and WikiText-2. To check whether this may relate to style (only SWBD is spoken style) and average sentence length (sentences on Switchboard are the shortest on average), we experiment with Transformer-based LMs on WSJ. The proposed approach reduces perplexity from 71.5 to 65.6 on test set (eval92), compared with a baseline Transformer LM. The improvement is relative 8.2% and in similar range with gains on WikiText-2. And this results in 0.1 absolute WER reduction from 1.5 to 1.4 on the same test set by N-best rescoring. While no conclusions can be drawn yet, experiments show that the proposed model perform better on written-style text which usually has longer average sentence length.

5.3. Analysis of Impact on Rare Words

One desired feature of the proposed model is that it may predict rare words better than LSTMs. To verify this, we further examine LM performance on the test set of each dataset. We split the vocabulary of each corpus into 10 buckets based on word frequencies in training data such that we get a roughly equal number of test tokens in each bucket. We then compute the differences between the test cross entropy of the proposed model and the LSTM baseline on words in each bucket for each dataset. Figure 2 shows the results on WikiText-2. As expected, larger reductions in cross-entropy are observed from the proposed model on rare words. Similar trends are seen on PTB and SWBD, though the overall perplexity improvement on SWBD is marginal. Figures for them are omitted due to page limits.

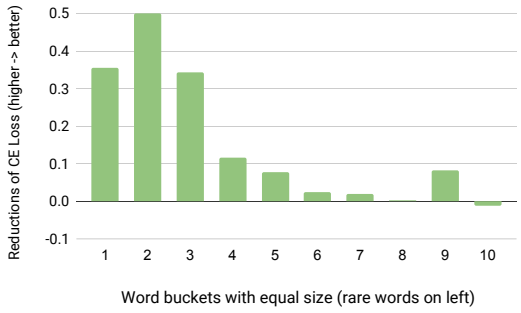

Figure 2: *Cross-entropy reduction from the proposed model w.r.t an LSTM on WikiText-2.*

5.4. Rescoring Evaluation and Analysis on SWBD

As Transformers show similar perplexity gains as LSTMs on SWBD, we only experiment with LSTMs for N-best rescoring. WERs on the full HUB5'00 evaluation set (Eval'00), the SWBD subset (SWB), and Callhome subset (CH) are in Table 6. "State-carry" means when scoring a hypothesis for current utterance, the initial hidden state is copied from the last hidden state of the best hypothesis for the previous utterance, instead of being zero initialized. WER improvements by the LSTM with "state-carry" in Table 6 indicate that cross-sentence context is useful. Similar observations are presented in [30]. If not specified with "w/o state-carry", models are evaluated in the state-carry way.

To investigate the effect on WERs of rare words, we conduct a similar analysis as Section 5.3 does. Words with errors on

Table 6: *WERs by N-best rescoring with baselines and the proposed model on SWBD.*

Model	Eval'00	SWB	CH
Kaldi RNNLM (w/o state-carry)	11.3	7.5	15.0
LSTM (w/o state-carry)	11.2	7.3	15.1
LSTM	10.9	7.1	14.5
PSMM	10.9	7.1	14.6
LSTM + Neural Cache	10.9	7.2	14.5
LSTM + Proposed	**10.8**	**7.1**	**14.4**

Eval'00 (<5000) are divided into 5 buckets based on frequency. Relative WER reductions on words in these buckets in Figure 3 show that the proposed model improves performance on both relatively rare words and very frequent ones. As expected, some rare words occur within the context window, for example, *masters* and *offered*, are recognized correctly. Decoded output also shows that frequent words such as *train*, *short*, and *were* are correctly recognized. Though the overall WER improvement by the proposed model is marginal, correctly recognizing relatively rare words plays an important role in user experience of ASR-based products or service.

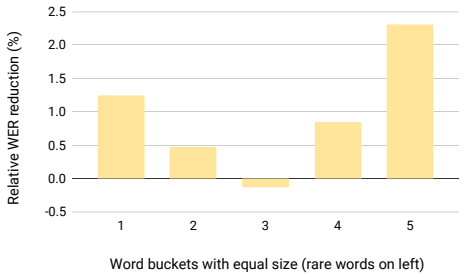

Figure 3: *Relative WER reduction on by the proposed model w.r.t an LSTM on SWBD.*

We also notice the proposed model sometimes introduces errors on words that are wrongly recognized in first pass decoding. A possible reason is that the supervision vectors for pointer components are from decoded hypotheses and hence may contain errors. We verify this by using test transcription in rescoring and observe a further 0.1 absolute WER reduction on Eval'00 of SWBD. The mismatched condition between training and evaluation is a common issue for the proposed approach, PSMM, and neural cache model. To alleviate the mismatch, word level confidence scores and error adaptive training approaches could be considered.

6. Conclusion and Future Work

In this work, we propose a cache-inspired pointer mechanism for neural LMs to improve the capacity of modeling long-range dependency and better predict rare words. It can be applied to both RNN- and Transformer-based models. Perplexity evaluation show that the proposed approach generally outperforms LSTM and PSMM and is more effective on rare words. Rescoring with the proposed model on SWBD and WSJ gives marginal WER improvements. Analysis shows that the mismatch between training and rescoring conditions (i.e. potentially incorrect histories) may make it challenging for both the proposed model and models with attention-based pointer network to achieve large overall WER reductions. Future work is therefore focused on methods that can mitigate the mismatch issue.

7. References

[1] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. of Interspeech*, 2010.

[2] X. Chen, X. Liu, M. J. Gales, and P. C. Woodland, "Recurrent neural network language model training with noise contrastive estimation for speech recognition," in *Proc. of ICASSP*, 2015.

[3] H. Xu, K. Li, Y. Wang, J. Wang, S. Kang, X. Chen, D. Povey, and S. Khudanpur, "Neural network language modeling with letter-based features and importance sampling," in *Proc. of ICASSP*, 2018.

[4] U. Khandelwal, H. He, P. Qi, and D. Jurafsky, "Sharp nearby, fuzzy far away: How neural language models use context," in *Proc. of ACL*, 2018.

[5] C. Chelba, M. Norouzi, and S. Bengio, "N-gram language modeling using recurrent neural network estimation," *arXiv preprint arXiv:1703.10724*, 2017.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. of NeurIPS*, 2017.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proc. of NAACL*, 2019.

[8] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[9] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," in *Proc. of ACL*, 2019.

[10] K. Irie, A. Zeyer, R. Schlüter, and H. Ney, "Language modeling with deep transformers," in *Proc. of Interspeech*, 2019.

[11] K. Li, Z. Liu, T. He, H. Huang, F. Peng, D. Povey, and S. Khudanpur, "An empirical study of transformer-based neural language model adaptation," in *Proc. of ICASSP*, 2020.

[12] R. Lau, R. Rosenfeld, and S. Roukos, "Trigger-based language models: A maximum entropy approach," in *Proc. of ICASSP*, 1993.

[13] K. W. Church, "Empirical estimates of adaptation: the chance of two noriegas is closer to p/2 than p^2," in *Proc. of COLING*, 2000.

[14] R. Kuhn and R. De Mori, "A cache-based natural language model for speech recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, no. 6, pp. 570–583, 1990.

[15] K. Li, H. Xu, Y. Wang, D. Povey, and S. Khudanpur, "Recurrent neural network language model adaptation for conversational speech recognition." in *Proc. of Interspeech*, 2018.

[16] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Proc. of NeurIPS*, 2015.

[17] J. Gu, Z. Lu, H. Li, and V. O. Li, "Incorporating copying mechanism in sequence-to-sequence learning," in *Proc. of ACL*, 2016.

[18] C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio, "Pointing the unknown words," in *Proc. of ACL*, 2016.

[19] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," in *Proc. of ICLR*, 2017.

[20] E. Grave, A. Joulin, and N. Usunier, "Improving neural language models with a continuous cache," *Proc. of ICLR*, 2017.

[21] B. Krause, E. Kahembwe, I. Murray, and S. Renals, "Dynamic evaluation of neural sequence models," *Proc. of ICML*, 2018.

[22] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing lstm language models," in *ICLR*, 2018.

[23] C. Gong, D. He, X. Tan, T. Qin, L. Wang, and T.-Y. Liu, "Frage: Frequency-agnostic word representation," in *Proc. of NeurIPS*, 2018.

[24] C. Wang, M. Li, and A. J. Smola, "Language models with transformers," *arXiv preprint arXiv:1904.09408*, 2019.

[25] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *Proc. of ASRU*, 2011.

[26] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohammadi, and S. Khudanpur, "Semi-orthogonal low-rank matrix factorization for deep neural networks." in *Proc. of Interspeech*, 2018.

[27] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for asr based on lattice-free mmi." in *Proc. of Interspeech*, 2016.

[28] T. Mikolov and G. Zweig, "Context dependent recurrent neural network language model," in *Proc. of SLT*, 2012.

[29] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[30] K. Irie, A. Zeyer, R. Schlüter, and H. Ney, "Training language models for long-span cross-sentence evaluation," in *Proc. of ASRU*, 2019.