OOV RECOVERY WITH EFFICIENT 2ND PASS DECODING AND OPEN-VOCABULARY WORD-LEVEL RNNLM RESCORING FOR HYBRID ASR

[1,2]*Xiaohui Zhang,*[2]*Daniel Povey,* [2]*Sanjeev Khudanpur*

[1]Facebook AI, US
[2] Center for Language and Speech Processing & Human Language Technology Center of Excellence
The Johns Hopkins University, Baltimore, MD 21218, US

xiaohuizhang@fb.com, dpovey@gmail.com, khudanpur@jhu.edu

ABSTRACT

In this paper, we investigate out-of-vocabulary (OOV) word recovery in hybrid automatic speech recognition (ASR) systems, with emphasis on dynamic vocabulary expansion for both Weight Finite State Transducer (WFST)-based decoding and word-level RNNLM rescoring. We first describe our OOV candidate generation method based on a hybrid lexical model (HLM) with phoneme-sequence constraints. Next, we introduce a framework for efficient second pass OOV recovery with a dynamically expanded vocabulary, showing that, by calibrating OOV candidates' language model (LM) scores, it significantly improves OOV recovery and overall decoding performance compared to HLM-based first pass decoding. Finally we propose an open-vocabulary word-level recurrent neural network language model (RNNLM) re-scoring framework, making it possible to re-score ASR hypotheses containing recovered OOVs, using a single word-level RNNLM ignorant of OOVs when it was trained. By evaluating OOV recovery and overall decoding performance on Spanish/English ASR 'tasks, we show the proposed OOV recovery pipeline has the potential of an efficient open-vocab word-based ASR decoding framework, with minimal extra computation versus a standard WFST based decoding and RNNLM rescoring pipeline.

***Index Terms*—** OOV recovery, RNNLM rescoring, dynamic vocabulary

1. INTRODUCTION

As a human language evolves, there are always new words occurring. Therefore it's not possible to cover all words in a language with a closed vocabulary. In a conventional hybrid ASR system, a pronunciation lexicon with a fixed vocabulary is usually used. In test phase, OOV words in test utterances can't be recognized correctly. And the recognizer will output acoustically similar in-vocabulary (IV) words as substitutions, and the recognition of surrounding in-vocabulary words may also be affected. This is harmful to readability of the ASR hypotheses, especially when the OOVs are named entities like human/place names, which could be important keywords given a certain context. Recent research on end-to-end (E2E) ASR with graphemes or word-pieces as modeling units has shown that it's possible to get rid of an expert-curated pronunciation lexicon [1], enabling open-vocabulary ASR. However its relative advantage over conventional lexicon-based ASR highly depends on the language and amount of data: for irregularly spelled languages like English, grapheme/word-piece-based ASR may only work well given a large amount of training data, e.g. thousands of hours. For academic English datasets with less than 1000 hours training data like Switchboard, Librispeech and Wall Street Journal (WSJ), it has been consistently observed that using a lexicon gives better performance[2, 3, 4]. Therefore, addressing the OOV detection/recovery problem for conventional lexicon-based ASR is still an important task, which we want to address here.

2. RELATED WORK

In order to recover OOVs, we need to first detect OOVs. The first type of approaches model OOVs implicitly with sub-word units, and detect OOVs either by finding inconsistency between sub-word (e.g. phone) and word recognition results via two pass decoding [5, 6, 7], or building a hybrid LM combining word and sub-word units (e.g. word-pieces) [8, 9, 10, 11, 12, 13]. Labeled data are usually required for building the OOV classifier. The second type of approaches model OOVs explicitly by a generic word model with a specific structure, e.g. a phonemic language model, to "absorb" OOVs during decoding [14, 15, 16, 17, 18]. The generic word model is tagged by an OOV symbol, e.g. <unk> in the lexicon and language model (LM), and is usually combined with the pronunciation lexicon, as a hybrid lexical model (HLM). This approach models OOVs explicitly and make it straightforward to conduct OOV recovery as a downstream task. So we'll follow it for OOV detection and candidate generation.

Given OOV candidates as strings of sub-words (e.g.

Thanks to Vimal Manohar and Chunxi Liu for insightful discussions, and Geoffrey Zweig and Yatharth Saraf for supports on this work

phonemes) from hybrid decoding with HLM, the next step is OOV recovery. Besides the basic approach using a phoneme-to-grapheme (P2G) model [18, 16] to recover OOVs' spellings, research efforts along two streams have been made to improve recovery performance. In the first stream, it's shown in [13, 19, 16] that clustering OOV candidates with linguistic/acoustic-based metrics helps OOV recovery, by more robustly dealing with different instances of an OOV word. However there's a basic assumption that there are many frequently recurring OOVs so that it's applicable to cluster OOV candidates. In our work we mostly focus on recovering infrequent/rare OOVs, and thereby we didn't pursue this direction. Instead we follow the second stream, improving OOV recovery by better estimating LM scores of OOV candidates in test phase. On one hand, [20, 21] investigated estimation of LM scores for OOVs given their ground-truth spelling from extra corpus, where relevant statistics can be collected. Our research will focus on OOV LM probability estimation without relying on oracle spellings. On the other hand, recently [22] proposed a hierarchy approach to first score hypotheses with a character RNNLM, and then use word/character RNNLMs to re-score in-vocabulary (IV)/OOV words separately. [1] Instead of combining two word/character RNNLMs, which is computationally expensive, we aim at extending the vocabulary of a single word RNNLM at test phase to enable re-scoring hypotheses containing OOVs.

3. OOV CANDIDATE GENERATION USING HYBRID LEXICAL MODEL (HLM)

A WFST-based decoder, e.g. the one used in Kaldi [23], performs decoding on a search space constrained by a composed graph $H \circ C \circ L \circ G$, where H maps acoustic modeling units to context-dependent phones; C represents the context-dependency; L presents the pronunciation lexicon, mapping phones to words. Following [14, 16, 15], we use the hybrid lexical model (HLM) to model OOVs in the lexicon. The first step is to train a phonemic language model on all pronunciation entries from the lexicon. Then, we compile this PLM into a finite state acceptor (FSA), denoted as G_p. However, G_p two undesirable properties. 1. It can generate phoneme sequences with any length, and thereby allows OOV candidates with a single phone pronunciation to occur in decoding lattices, which is not optimal. 2. It's trained on phonemes without word-position-dependency to keep it compact, while we need word-position-dependency to represent OOV pronunciations to be consistent with other lexicon entries. We solve these two problems by composing the G_p with a separate "phoneme-sequence-constraint" FST, denoted as C_p, encoding the allowed minimum length (e.g. 2) of generated sequences; and adding word-position-dependency. See Figure 1 for an example.

[1]Although using a character LM alone solves the OOV problem, it's generally believed that character LM under-performs relative to word LMs for languages with a small character vocabulary like English [22].

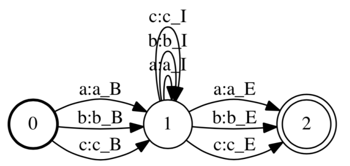

Fig. 1. An example "phoneme-sequence-constraint" FST accepting only phoneme sequences whose length ≥ 2, and adding word-position-independent markers ("B", "E", "I" means at the beginning/end of or inside a word) to phonemes

After creating $G_p \circ C_p$, we build the lexicon L as usual, and replace the OOV symbol, e.g. <unk>'s pronunciation with $G_p \circ C_p$ using OpenFST's `fstreplace` operation. This gives the hybrid lexical model (HLM), and we'll have a hybrid decoding graph: $H \circ C \circ (L \cup (G_p \circ C_p)) \circ G$. The weight of the OOV model in the graph is controlled by the unigram probability of <unk> in G. After generating lattices in a standard WFST decoding pipeline, and aligning all arcs with word boundaries, OOV instances will occur as arcs labeled by <unk> with a pronunciation given by the HLM[2]. A P2G model can then be applied to recover OOV candidates' spelling.

4. EFFICIENT 2ND PASS DECODING WITH A DYNAMIC VOCABULARY

Having obtained OOV candidates by hybrid decoding with HLM, we now focus on improving OOV recovery performance by 2nd pass decoding. Our motivation is to re-generate lattices better accounting for OOVs by calibrating OOV candidates' LM scores utilizing their pronunciation, spelling, empirical frequency, and the overall empirical OOV rate, etc., which were unavailable during 1st pass decoding. The first challenge is how to build the OOV grammar G_{oov}. First, we need to assign a proper unigram probability mass[3] to the OOV grammar. Here we assign it by the empirical OOV rate, i.e. $\frac{\#.\ <unk>\ arcs\ from\ lattices}{\#.\ all\ arcs\ from\ lattice}$ boosted by a tunable scalar working as a "prior" of G_{oov}). By this mechanism, we achieve our goal of calibrating OOVs' weight according to 1st pass decoding results. Then the left problem is to estimate the distribution of OOV candidates. Here we'll empirically explore following estimators: **Phonemic LM (PLM) scores**: This is the most convenient option since we can reuse the PLM used to build HLM in 1st pass decoding; **Character LM scores**: This option is more informative than PLM scores for languages which are irregularly spelled/has a large grapheme set, and we can usually train a strong neural character LM; **Empirical frequency**, which is number of arcs of a particular OOV

[2]We determinize lattices at both word+phone level and word level, by which we keep 1-best path of each local phone lattice representing an <unk> instance. Note that one OOV could have multiple <unk> instances.

[3]Here we only estimate OOV candidates' unigram prob., since we focus on infrequent OOVs so that it's hard to robustly estimate contextual statistics.

candidate found in lattices, divided by the total number of arcs; **Uniform distribution**, meaning we just assign a constant probability to all OOV candidates. This is potentially a bad option, which will be investigated as a baseline. Given G_{oov}, the challenge left is how to efficiently do 2nd pass decoding. The trivial approach is to insert it into the regular LM G, expand the lexicon L accordingly, and rebuild the decoding graph, which is very expensive. Another solution is to utilize a dynamic decoder which compiles the decoding graph on-the-fly with context dependency and look-ahead [24, 25], which loses the speed advantage of a static decoder. The method we adopt is Kaldi's Grammar-decoding framework designed for situations where we need to dynamically expand a vocabulary efficiently at run-time. It utilizes a pre-compiled generic static decoding graph as usual. During 2nd pass decoding, we build a small decoding graph $H \circ C \circ L_{oov} \circ G_{oov}$ where L_{oov} is the lexicon of recovered OOVs built with P2G. Then during decoding, we stitch it together with the generic decoding graph dynamically. The framework is specific to models with left-biphone phonetic context. This makes it feasible to enumerate the entry points of each sub-graph (one per left-context phone). This solution keeps the speed advantage of a static decoder, while avoiding re-building the whole decoding graph at run-time.

5. OPEN-VOCABULARY RNNLM RE-SCORING

Normally, a word-level RNNLM couldn't inference on hypotheses containing OOVs. To tackle this challenge, we utilize two design features of Kaldi-RNNLM[26] enabling dynamic vocabulary expansion at test phase: (1). **Subword embedding**. The word embedding $W_{N\times M}$ (N: vocab size; M: embedding dimension) is factorized as the product of a sparse feature matrix $F_{N\times K}$ (K is the number of features, including sub-word features like Bag-of-Words (BoW) of character n-grams) and a dense subword embedding matrix $Y_{K\times M}$ trained jointly with the neural LM. (2). **Input/Output embedding tying**, i.e. $W_{N\times M}$ is shared by both input/output layers of the RNN [27]. Specifically, assuming L recovered OOVs encountered at test phase, we first extract their subword features and augment the original $F_{N\times K}$ to get $F'_{(N+L)\times K}$. Then we multiple it with the original subword embedding $Y_{K\times M}$ to get the augmented word embedding $W'_{(N+L)\times M}$ and use it to replace $W_{N\times M}$ at both input/output layers [4]. Then this RNNLM can inference on sentences containing recovered OOVs. In order to enable lattice rescoring, we also need to combine (by `fstreplace`) the n-gram LMs G and G_{oov} which were used in 2nd pass Grammar-decoding, into a single FST, for subtracting old LM scores from lattices. Then we can re-score lattices/n-best lists as usual.

[4]Kaldi-RNNLM optionally adds a normalization step at output during inference, so that probabilities sum up to one after vocabulary expansion.

6. EXPERIMENTS

For the performance metric for OOV recovery, we first align each reference OOV token with its max-overlapped hypothesis token, and then compute word error rate WER (defined as 1-recovery rate) and character error rate (CER) of all OOV tokens. Besides, we'll measure overall WER/CER on utterance level since we also aim at improving overall decoding performance.

6.1. Spanish experiments on *Heroico*

We first evaluate the proposed OOV recovery pipeline on a Spanish ASR task *Heroico* (LDC2006S37), since Spanish is a regularly spelled language, creating a easier evaluation condition ruling out potential errors from P2G. Its training set contains 11hrs read & free-response answer speech. Two test sets contain the same texts read by native/non-native speakers, all non-seen in training, with duration of 1h/1.2h respectively. To simulate OOVs, we randomly removed 45K infrequent words (count < 10) from the 91K vocabulary. The OOV rate w.r.t. the limited lexicon on both test sets is 20.6%. We then use the limited lexicon to train a G2P model and a bi-gram PLM used for OOV recovery. Then we proceed with the proposed 1st pass decoding with HLM, 2nd pass grammar decoding and then open-vocab RNNLM re-scoring. Besides, a baseline is obtained by decoding with the same limited vocabulary, 1st pass decoding with n-gram LM and then regular RNNLM rescoring, without OOV recovery; and oracle results are obtained by the same decoding procedure as the baseline, but using the original lexicon/LM containing all simulated OOVs. For all experiments, the same neural acoustic model (TDNN-F [28]) is used.

First, we present an overview of OOV recovery and overall decoding performance in Table 1. On all metrics, 2nd pass decoding improves over 1st pass decoding consistently, more significantly on OOV WER/CER. Open-vocab RNNLM rescoring also consistently helps on all metrics, and the relative gain it brings is always larger on top of 2nd pass decoding lattices than 1st pass, confirming 2nd pass decoding provides lattices of higher quality by calibrating OOV candidates' LM scores. Overall, the best system "2nd pass + RNNLM" fills 20%-50% of the gap between the baseline and oracle results on overall WER/CER, 30%-50% on OOV WER, and 80% on OOV CER. The larger gain on OOV CER than OOV WER means recognizing OOVs exactly is much harder than recognizing them approximately. Also the gain for each metric is smaller on "Nonnative" set, since the speech by Nonnative speakers are accented so that the quality of OOV candidates from phonemic decoding is worse.

Next, we investigate different OOV unigram probability estimators in 2nd pass decoding, with results on OOV CER (the most informative metric for OOV recovery) presented in Table 2. As mention before, we investigate uniform distribution ("Unif."), empirical frequency ("$\hat{f}$"), and phonemic/character LM scores. For the phonemic LM (PLM), we

Test set	LM	Overall WER				Overall CER				OOV WER				OOV CER			
		Baseline	pass 1	pass 2	Oracle	Baseline	pass 1	pass 2	Oracle	Baseline	pass 1	pass 2	Oracle	Baseline	pass 1	pass 2	Oracle
Native	n-gram	29.7	27.6	24.1	5.9	10.3	7.6	6.6	1.9	100	90.5	68.3	7.3	124.8	36.9	26.4	2.4
	RNNLM	29.1	24.5	19.7	4.95	10.0	6.9	5.5	1.6	100	78.2	55.6	6.1	125.2	31.3	21.2	2.1
Non-Native	n-gram	34.6	33.7	31.7	11.6	14.5	13.1	11.9	4.97	100	95.5	83.0	14.4	124.0	43.1	36.5	6.7
	RNNLM	33.7	32.1	28.5	10.1	13.9	11.9	10.4	4.31	100	90.8	70.4	12.9	124.6	39.9	29.3	6.3

Table 1. Performance overview of 2nd pass decoding & open-vocab RNNLM rescoring on the *Heroico* Spanish ASR task

directly take the n-gram PLM used to build HLM. For the character LM, we train a character RNNLM ("CharRnnlm") on all in-vocab words. We can see that uniform distribution and empirical frequency perform very bad, mainly because they are not able to assign low scores to long and incorrect OOV candidates, resulting in lots of insertion errors. Besides, "CharRnnlm" outperforms "PLM" a bit, mainly because of the power of neural language modeling, since a Spanish word's pronunciation/spelling contain almost the same information. Furthermore, we tried composing CharRnnlm scores with $\hat{f}$ [5], and found it helps a bit, which was chosen as the best estimator for this task. Note that the decisions of using phonemic/character scores and composition with $\hat{f}$ should be empirically decided for each language/task, e.g. we found composition with $\hat{f}$ only helps when the OOV rate is high.

Test set	Unif.	$\hat{f}$	PLM	CharRnnlm	CharRnnlm w/ $\hat{f}$
Native	100.8	96.0	34.1	26.4	24.3
Nonnative	115.4	118.3	42.5	36.5	34.3

Table 2. OOV CER (2nd pass decoding + RNNLM rescoring) with different OOV LM probability estimators

6.2. English experiments on *Librispeech*

Now we switch to the more challenging but practical condition: *Librispeech* English ASR task. Here the language is irregularly spelled, and we use the official 200K vocab so that OOV rates on *test-clean/other* are very low (0.4%-0.5%), and the OOVs are harder to recognize, since they're mostly foreign words/name entities. Results are presented in Table 3. All results are after RNNLM rescoring, and the "OOV recovery" row stands for the best 2nd pass decoding + open-vocab RNNLM rescoring system. Here we add two extra evaluation conditions. In practice, during test time, we sometimes know the spelling (e.g. from extra texts) of OOVs in test speech. We simulate this condition by adding all oracle OOVs from each test set into the lexicon with (up to top-3) G2P generated pronunciations before decoding, hence the "+Oracle Spelling" condition[6]. Given oracle OOV spellings, we want to show the proposed pipeline could still improve performance, by learning OOV pronunciations from acoustic evidence, hence the "+Acoustic Evidence" condition. Basically, we collect all OOV pronunciations from 1st pass hybrid decoding. For each pronunciation, we apply P2G and take top-N (which is large, like 100) spellings, choose pronunciations with at least one possible spelling matching an oracle OOV, and add them to the lexicon as extra pronunciations learned from acoustic evidence for those oracle OOVs.

Looking at first two rows in Table 3, as expected, the gain by OOV recovery is much smaller than in the previous Spanish task, on all metrics. Looking at the last two rows, it shows we can achieve consistent gains on all metrics, more notably on OOV WER/CERs, by utilizing OOVs' pronunciations learned from acoustic evidence. We picked several oracle OOVs only correctly recognized at the "+Acoustic evidence" condition, and found the pronunciations from acoustic evidence are indeed better than from G2P (mostly differ in vowels only), e.g. STUTELEY: `S T UW1 T L IY0` (G2P) vs. `S T AH1 T L IY0` (acoustic evidence). Besides, the gain is always larger on *test-clean* than *test-other*, again confirming performance of the proposed pipeline highly relies on the audio quality.

	Overall WER		OOV WER		OOV CER	
Test set	*clean*	*other*	*clean*	*other*	*clean*	*other*
Baseline	2.84	7.38	100	100	46.01	53.11
OOV recovery	2.75	7.41	94.55	98.58	37.44	50.13
+Oracle Spelling	2.56	7.18	50.91	68.44	24.08	39.12
+Acoustic Evidence	**2.49**	**7.14**	**42.27**	**64.18**	**19.52**	**37.23**

Table 3. Performance of the proposed pipeline on *Libripseech*

7. CONCLUSION

In this paper we have proposed an OOV recovery pipeline tightly integrated to a standard WFST decoding and RNNLM re-scoring framework, with both OOV recovery and overall decoding performance improved on Spanish/English ASR tasks, by enabling OOV LM score estimation, efficient 2nd pass decoding with a dynamic vocabulary, and open-vocab word RNNLM re-scoring. Minimal extra computational overhead has made it a practical framework for word-based open-vocabulary ASR decoding. Besides, we showed that even given oracle OOV spellings, the proposed framework can bring further gain by learning OOV pronunciations from acoustic evidence.

[5]We tried interpolation with tuned weights and element-wise production + re-normalization. The later works better, giving a sharper distribution.

[6]Note that the LM scores of oracle OOVs are still estimated the same way as the OOV recovery pipeline, since they are not in the original LM.

8. REFERENCES

[1] T. Sainath, R. Prabhavalkar, S. Kumar, et al., "No need for a lexicon? evaluating the value of the pronunciation lexica in end-to-end models," *Proc. ICASSP*, 2017.

[2] T. Likhomanenko, G. Synnaeve, and R. Collobert, "Who needs words? lexicon-free speech recognition," *Proc. INTERSPEECH*, 2019.

[3] H. Hadian, H. Sameti, D. Povey, and S. Khudanpur, "Flat-start single-stage discriminatively trained hmm-based models for asr," *IEEE/ACM TASLP*, 2018.

[4] C. Lscher, E. Beck, K. Irie, M. Kitza, W. Michel, A. Zeyer, R. Schlter, and H. Ney, "Rwth asr systems for librispeech: Hybrid vs attention - w/o data augmentation," in *Proc. INTERSPEECH*, 2019.

[5] C. White, G. Zweig, L. Burget, P. Schwarz, and H. Hermansky, "Confidence estimation, oov detection and language id using phone-to-word transduction and phone-level alignments," in *Proc. ICASSP*, 2008.

[6] H. Lin, J. Bilmes, D. Vergyri, and K. Kirchhoff, "Oov detection by joint word/phone lattice alignment," in *Proc. ASRU*, 2007.

[7] S. Kombrink, L. Burget, P. Matějka, M. Karafiát, and H. Hermansky, "Posterior-based out of vocabulary word detection in telephone speech," in *Proc. INTERSPEECH*, 2009.

[8] D. Klakow, G. Rose, and X. Aubert, "Oov-detection in large vocabulary system using automatically defined word-fragments as fillers," *Proc. EUROSPEECH*, 1999.

[9] A. Yazgan and M. Saraclar, "Hybrid language models for out of vocabulary word detection in large vocabulary conversational speech recognition," *Proc. ICASSP*, 2004.

[10] M. Bisani and H. Ney, "Open vocabulary speech recognition with flat hybrid models," in *Proc. EUROSPEECH*, 2005.

[11] A. Rastrow, A. Sethy, and B. Ramabhadran, "A new method for oov detection using hybrid word/fragment system," in *Proc. ICASSP*, 2009.

[12] H. Kuo, E. Kislal, L. Mangu, H. Soltau, and T. Beran, "Out-of-vocabulary word detection in a speech-to-speech translation system," in *Proc. ICASSP*, 2014, pp. 7108–7112.

[13] L. Qin and A. Rudnicky, "Finding recurrent out-of-vocabulary words.," *Proc. INTERSPEECH*, 2013.

[14] I. Bazzi and J. Glass, "Modeling out-of-vocabulary words for robust speech recognition," *Proc. ICASSP*, 2000.

[15] I. Szöke, "Hybrid word-subword spoken term detection," *BRNO University of Technology Department of Computer Graphics and Multimedia*, 2010.

[16] E. Egorova and L. Burget, "Out-of-vocabulary word recovery using fst-based subword unit clustering in a hybrid asr system," in *Proc. ICASSP*, 2018.

[17] M. Basha Shaik, D. Rybach, S. Hahn, R. Schlüter, and H. Ney, "Hierarchical hybrid language models for open vocabulary continuous speech recognition using wfst," in *SAPA-SCALE Conference*, 2012.

[18] S. Kombrink, M. Hannemann, L. Burget, and H. Heřmanskỳ, "Recovery of rare words in lecture speech," in *International Conference on Text, Speech and Dialogue*. Springer, 2010, pp. 330–337.

[19] M. Hannemann, S. Kombrink, M. Karafiát, and L. Burget, "Similarity scoring for recognizing repeated out-of-vocabulary words," in *Proc. INTERSPEECH*, 2010.

[20] I. Illina and D. Fohr, "Out-of-vocabulary word probability estimation using rnn language model," *LTC*, 2017.

[21] L. Orosanu and D. Jouvet, "Adding new words into a language model using parameters of known words with similar behavior," *Procedia Computer Science*, vol. 128, pp. 18–24, 2018.

[22] Takaaki Hori, Shinji Watanabe, and John R Hershey, "Multi-level language modeling and decoding for open vocabulary end-to-end speech recognition," in *Proc. ASRU*, 2017.

[23] D. Povey, A. Ghoshal, G. Boulianne, et al., "The kaldi speech recognition toolkit," *Proc. ASRU*, 2011.

[24] C. Allauzen, M. Riley, and J. Schalkwyk, "A generalized composition algorithm for weighted finite-state transducers," *Proc. INTERSPEECH*, 2009.

[25] P. Dixon, C. Hori, and H. Kashioka, "A specialized wfst approach for class models and dynamic vocabulary," in *Proc. INTERSPEECH*, 2012.

[26] H. Xu, K. Li, et al., "Neural network language modeling with letter-based features and importance sampling," in *Proc. ICASSP*, 2018.

[27] O. Press and L. Wolf, "Using the output embedding to improve language models," *Proc. EACL*, 2017.

[28] D. Povey, G. Cheng, et al., "Semi-orthogonal low-rank matrix factorization for deep neural networks," in *Proceedings of INTERSPEECH*, 2018.