IMPROVEMENTS TO N-GRAM LANGUAGE MODEL USING TEXT GENERATED FROM NEURAL LANGUAGE MODEL

Masayuki Suzuki, Nobuyasu Itoh, Tohru Nagano, Gakuto Kurata, and Samuel Thomas

IBM Research AI

ABSTRACT

Although neural language models have emerged, n-gram language models are still used for many speech recognition tasks. This paper proposes four methods to improve n-gram language models using text generated from a recurrent neural network language model (RNNLM). First, we use multiple RNNLMs from different domains instead of a single RNNLM. The final n-gram language model is obtained by interpolating generated n-gram models from each domain. Second, we use subwords instead of words for RNNLM to reduce the out-of-vocabulary rate. Third, we generate text templates using an RNNLM for template-based data augmentation for named entities. Fourth, we use both forward RNNLM and backward RNNLM to generate text. We found that these four methods improved performance of speech recognition up to 4% relative in various tasks.

Index Terms— n-gram, RNNLM, interpolation, subword, template

1. INTRODUCTION

n-gram language models (LMs) are still used for speech recognition systems such as hybrid systems [1, 2, 3, 4], so improving n-gram LMs using neural LMs is still an important research topic. There are many approaches to leverage neural LMs for n-gram LMs [5, 6, 7]. One simple and promising approaches is to generate text from a recurrent neural network language model (RNNLM) and build n-gram LMs using the generated text [8, 9, 10]. This paper proposes four methods to improve n-gram LM using text generated from RNNLM.

We detail the four methods in Section 2. Section 3 describes related works. We evaluate the effectiveness of the proposed methods on variety of tasks in Section 4. Section 5 concludes this paper.

2. PROPOSED METHODS

2.1. Domain wise generation

Ideal training data for LMs is matched domain and sufficiently large data. In practice, training data consists of multiple corpora, which have different domains and sizes. Web crawling data tends to be large enough, but has a domain different from a target domain because it is written text. Matched domain data is often not large enough. The standard method to make an n-gram LM using these data is to interpolate multiple n-gram LMs from each domain [11]. Interpolation weights are estimated using an EM algorithm with a development dataset.

Inspired by the method to make an interpolated n-gram LM from multiple corpora, we propose using multiple RNNLMs, generating multiple n-gram LMs, and interpolating them to make a final n-gram LM. Fig. 1 shows a flowchart of this method. First, we do balance sampling from datasets to make a single balanced corpus as (1) in

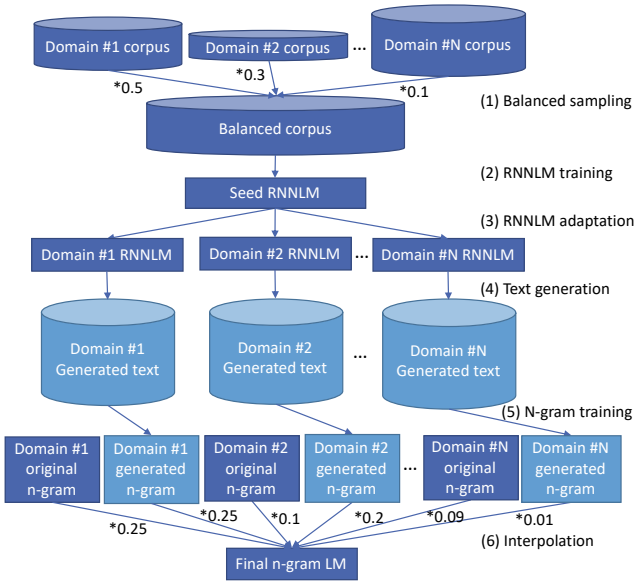

Fig. 1. Flowchart to make an n-gram LM by domain wise text generation of RNNLMs. Domain #k corpus is used to adapt domain #k RNNLM and to train domain #k original n-gram.

Fig. 1. We can reuse interpolation weight of the standard recipe to make an interpolated n-gram LM for the balance sampling. The balanced corpus is used to train a seed RNNLM as (2). We adapt the seed RNNLM by fine-tuning using each corpus as (3) [12] . We generate many texts using each adapted RNNLM as (4) and built n-gram LMs as (5). The final n-gram LM is obtained by interpolating both original n-gram LMs and generated n-gram LMs as (6).

2.2. Subword generation and word acquisition

Huang et al. [13] proposed using subwords for text generation of RNNLM. It performs superiorly for languages that have many compound words. In addition, we can obtain new vocabulary from its generated text.

One problem in word acquisition using subword RNNLM is that generated text can include unnatural words by combining an impossible pair of subwords. We propose using a large web corpus to filter out these unnatural words. For major languages, we can crawl plenty of text from the web. This kind of corpus includes many irrelevant words for a target domain, but almost all words are natural ones. Thus, words that frequently appear in a corpus generated by RNNLM of the target domain and appeared at least once in the web corpus can be good candidates for enhancing vocabulary.

2.3. Template generation

The error rate of named entities is an important metric for many practical applications of speech recognition [14, 15]. One practical way to improve error rates is to augment text using templates [16]. For example, from a template "I went to [city]" and list of cities, we can generate "I went to Tokyo" and "I went to New York".

One problem in the template-based data augmentation is that making large template data requires large effort. If template data is not large enough, normal words in the template repeatedly appeared in the augmented data. To mitigate this problem, we propose generating templates themselves using RNNLM. The RNNLM for template generation has special words such as [city] so that it can input and output templates. The RNNLM is trained using manually prepared templates as training data. We do transfer learning from the seed RNNLM in Fig. 1, except for parameters corresponding to the special words, to leverage general knowledge of the language. Trained template RNNLM possibly generates not only "I went to [city]" that appears in training data but also "I visited [city]" using the knowledge. After generating a large amount of templates, we replace these special words with specific examples to make a corpus to train an n-gram.

2.4. Backward generation

Although a bidirectional RNN generally performs superiorly to an unidirectional RNN, it is not straightforward to use future contexts for RNNLM because language modeling is a task to estimate future words [17, 18]. Instead of using a bidirectional model somehow, we propose using another backward RNNLM to generate text. Because text generation is inherently offline processing during training time, using backward RNNLM does not affect test time processing. We use both a forward RNNLM generated text corpus and a backward RNNLM generated text corpus to make two n-gram LMs for each domain. All the n-gram components are used to make the final n-gram LM as in Fig. 1.

Note that we can use not only backward RNNLM but also arbitrary architectures of neural LMs in this scheme [19, 20].

3. RELATED WORK

There are many methods that do not require n-gram LMs leveraged by neural LMs. Some use neural LMs directly for decoding [21, 22], and others require no external LMs [23, 24]. We stick, however, to n-gram LMs because current state-of-the-art speech recognition systems use n-gram LMs [1, 2, 3] and n-gram is easy to customize for commercial use cases where many customers frequently need to add new words and adapt the LM using small amounts of text data.

Arisoy et al. [6] and Adel et al. [7] proposed methods to convert a neural LM into an n-gram LM directly. On the other hand, we first generate text using a neural LM, then train an n-gram LM using the generated text. One merit of generating text is that it can use different word units for RNNLM and n-gram LMs. In addition, it can use arbitrary architectures of recently proposed neural LMs [19, 20].

Using a mixture of LMs of multiple domains is a widely accepted idea in language modeling [11, 25, 26]. We applied this idea for text generation by neural LMs.

Many papers have investigated the effectiveness of subwords for neural LM [27, 28]. Subword RNNLM was applied for text generation by Huang et al. [13]. Our contribution to this topic is to propose a filtering method for word acquisition as explained in Section 2.2.

Bidirectional RNNLMs have been investigated in [17, 18] for online processing. Xiong et al. [2] uses both forward and backward RNNLM for offline n-best rescoring of speech recognition results. Similar to the them, we propose using both forward and backward RNNLM for text generation.

4. EXPERIMENTS

We selected in-house datasets of Korean, Japanese, and English to evaluate our proposed methods. The Korean task was selected to see the effect of subword modeling. Because the unit we used for LM of Korean is relatively long, subword modeling was expected to have more positive results than for other languages. The Japanese task was an adaptation task to a specific domain where addresses and names are often uttered. This task was selected to evaluate template generation, which can effectively generate sentences including addresses and names. The English task was selected to evaluate our proposed method in a rich resource language.

4.1. Korean

The left half of Table 1 shows training data and its word counts. Five transcribed datasets and three web crawling datasets are used for training. We used four test datasets for evaluation: A, B, C, and X. Test set A, B, and C were sampled from the same domain corpus as that of training sets A, B, and C, respectively. Data from a domain of test set X was not included in training data. Test sets A, B, C, and X had 15K, 3K, 15K, and 3K words, respectively. To estimate interpolation weights of n-gram models, we also prepared 6K words for a development set. There was no overlap between training data, development data, and test data.

We decided to use Korean phrases, which are segmented by blank spaces in written form, as an unit of an n-gram LM. One reason for this was that we could make a pronunciation dictionary considering liaison, which mainly occurs within Korean phrases [29]. For subword modeling for RNNLM, we used the unigram language model algorithm implemented in sentencepiece[1] [28]. The subword vocabulary size was set to 8K.

RNNLM had a word embedding and one layer of gated recurrent unit (GRU) layer [30]. Dimension of the word embedding and the GRU layer were both 1024. For subword RNNLM, full 8K vocabulary was used for inputs and outputs. For word RNNLM, the top 30K words were used for inputs and outputs. Standard cross entropy criterion, an SGD algorithm, dropout were used to train these RNNLMs.

We built an interpolated n-gram LM following the flow in Fig. 1. For each RNNLM, we generated 1 billion words. We trained a 4-gram LM with modified Kneser-Ney smoothing [31]. Interpolation weights were estimated using an EM algorithm using the development data. As for data of news site F, we did not generate any text because the original data size was large enough.

As for an acoustic model (AM), we used a convolutional neural network (CNN) that has two convolution layers and five fully connected layers [32]. The size of the training data was 1K hours. The student-teacher framework was used to train the CNN using the VGG model as a teacher [33, 34].

Table 2 shows results. We tried not only our proposed flow shown in Fig. 1 but also using only a seed RNNLM to generate text and an n-gram for comparison. In addition, we tried both word

[1] https://github.com/google/sentencepiece

Table 1. Corpus name, word count, and estimated interpolation weights for Korean experiments. Weights #1 represents estimated interpolation weights for a baseline model. Weights #5 represents that of proposed method, which uses n-gram models of both original text and subword RNNLM generated text of multiple domains.

Corpus name	Word count	Weights #1	Weights #5		
			Original	Generated	Total
Transcribed data - domain A	771K	0.4312	0.0926	0.3459	0.4385
Transcribed data - domain B	638K	0.1601	0.0243	0.1365	0.1608
Transcribed data - domain C	2,666K	0.1581	0.0282	0.0773	0.1056
Transcribed data - domain D	998K	0.0185	0.0041	0.0048	0.0089
Transcribed data - domain E	9K	0.0016	0.0002	0.0623	0.0625
Web crawling - news site F	507,057K	0.1501	0.1256	n/a	0.1256
Web crawling - related site G	565K	0.0424	0.0075	0.0220	0.0295
Web crawling - related site H	219K	0.0379	0.0054	0.0632	0.0686
Total		1.0000	0.2879	0.7121	1.0000

Table 2. %CERs using different components for interpolated n-gram LMs for Korean task

	Components	Test A	Test B	Test C	Test X	Average
#1	8 original n-gram ...(*)	14.0	23.0	33.9	16.0	21.73
#2	(*) + 1 seed-word-RNNLM generated n-gram	13.8	22.4	33.9	15.8	21.48
#3	(*) + 7 word-RNNLMs generated n-gram	13.8	22.5	33.6	15.3	21.30
#4	(*) + 1 seed-subword-RNNLM generated n-gram	13.8	22.4	33.7	15.7	21.40
#5	(*) + 7 subword-RNNLMs generated n-gram	13.6	22.2	33.6	15.2	21.15

Table 3. %OOV rate and %CERs using different vocabulary for Korean task

		Voc size	OOV rate	Test A	Test B	Test C	Test X	Average
#6	Words in transcribed data ...(*)	77K	4.70	13.9	29.6	33.8	15.4	23.18
#7	(*) + Frequent words in web data	220K	4.37	13.8	25.0	33.8	15.6	22.05
#8	(*) + Frequent words in generated data	220K	3.42	13.7	22.4	33.4	15.2	21.18
#9 == #5	(*) + Combination	220K	3.41	13.6	22.2	33.6	15.2	21.15

RNNLM and subword RNNLM to see the effect of subword modeling. We use the character error rate (CER) for the performance metric instead of the word error rate (WER) because segmentation of Korean phrases are ambiguous. #1 was a baseline results without using RNNLM text generation. RNNLM text generation was effective because all results from #2 to #5 had better CERs than #1. Using multiple RNNLMs and generated n-gram models for each domain was better than using single seed RNNLM (#2 vs. #3, and #4 vs. #5). Using a subword unit for RNNLM text generation had better CERs than that of word RNNLM (#2 vs. #4, and #3 vs. #5).

The right half part of Table 1 shows the estimated interpolation weights of #1 and #5 in Table 2. Weights for generated n-gram models were larger than those of the original n-gram models. Because we could generate enough large data for n-gram LM, RNNLM text generation was effective, especially for domains where original data size was small. Weight for domain E was increased after adding generated n-gram LMs. Domain E data was relevant data but it had small weight because its corpus size was very small. Thanks to the RNNLM text generation, we could increase the text related to the target domain from a small corpus.

Next, we changed vocabulary and saw out-of-vocabulary (OOV) rates and CERs. We used the same data to train LMs for #5 in Table 2. Table 3 shows the results. #6 is a result using a 77K vocabulary that appears in the five transcribed datasets. #7, #8, and #9 added 143K words so that vocabulary size became 220K. Increasing vocabulary size improved OOV rates and CERs. Comparing #7 and #8, using the subword RNNLM generated corpus to add words was better than using web crawling data. As for #9, the top 100K frequent words in generated data and the next top 43K frequent words that also appeared in the web crawling corpora are used to filter out unnatural words generated by subword RNNLM. #9 slightly outperforms #8. We used this vocabulary for experiments for Table 2. #9 in Table 3 and #5 in Table 2 were the same experimental setting.

4.2. Japanese - address and name

This task involved adapting a general purpose n-gram LM to a specific domain n-gram LM. We interpolated a general purpose n-gram and n-gram LMs trained using adaptation data. Adaptation data, development data, and test data consisted of 10K, 4K, and 4K words, respectively. There was no overlap. This domain included many utterances that included addresses and names. From the training data, we manually made template data by replacing the addresses and names to special words, e.g., [First name]. This template datasets has 2K words and 315 special words. For each special word, we prepared a list of possible examples, e.g., Masayuki for [First name]. Table 4 shows special words used for the template data.

Subword modeling, RNNLM, n-gram LM, and AM setups were the same as in the Korean experiment but with different training data.

Table 5 shows the CERs. #10 shows results of the general purpose n-gram. Interpolating an n-gram using the adaptation data had large gain (#10 vs. #11). Then, we trained a seed-subword-RNNLM using a balanced corpus, adapted it using the adaptation data, generated texts from the adapted-subword-RNNLM, and built an n-gram LM. Interpolating this n-gram had better result (#11 vs. #12). Next, we generated 15M words of text using the template data repeatedly and built an n-gram. Adding this n-gram achieved better results (#12 vs. #13). Finally, we adapted the RNNLM using the template data and generated 50M words of templates as explained in Section 2.3. We generated 50M words of text replacing special words in gener-

Table 4. Special words, # of examples, and its counts in the template dataset. We separated whole Japanese addresses into four parts and prepared special words for each combination of these parts to represents possible colloquial variations[2].

Special word	# of examples	count
[Family name]	7,370	165
[First name]	5,595	66
[Address 1-2-3-4]	36,377	2
[Address 3-4]	35,709	4
[Address 4]	27,857	17
[Address 2-3]	27,220	4
[Address 2-3-4]	26,418	1
[Address 1-2-3]	13,655	1
[Address 3]	5,603	15
[Address 2]	4,989	23
[Address 1-2]	3,366	12
[Address 1]	93	5

Table 5. CERs using different components for interpolated n-gram LMs for Japanese task

	Components	%CER
#10	general purpose n-gram ... (1)	13.27
#11	(1) + adaptation data n-gram ... (2)	11.60
#12	(2) + RNNLM generated n-gram ... (3)	11.42
#13	(3) + Template generated n-gram ... (4)	11.34
#14	(4) + Template-RNNLM generated n-gram	11.12

Table 6. WERs using different components for interpolated n-gram LMs for English task. "RNNLM" means n-gram LM trained using generated text from the RNNLM. "backward" means n-gram LM trained using generated text from backward version of the RNNLM.

	Components	%WER
#15	baseline n-gram ... (*)	9.89
#16	(*) + seed-subword-RNNLM	9.84
#17	(*) + seed-subword-RNNLM + backward	9.82
#18	(*) + seed-word-RNNLM	9.78
#19	(*) + seed-word-RNNLM + backward	9.78
#20	(*)+ adaptation data n-gram ... (**)	9.40
#21	(**) + adapted-subword-RNNLM	9.28
#22	(**) + adapted-subword-RNNLM + backward	9.23
#23	(**) + adapted-word-RNNLM	9.21
#24	(**) + adapted-word-RNNLM + backward	9.20

ated templates, and built an n-gram. Adding this n-gram further improved performance (#13 vs. #14).

4.3. English

This task involved adapting an n-gram LM for broadcast news to a specific channel. The baseline broadcast news n-gram LM was trained using 5.8 billion words of text. Adaptation data, development data, and test data consisted of 2M, 13K, and 72K words, respectively. There was no overlap. We first trained a seed-RNNLM using the sampled data of the corpus for the baseline model. Then, we adapted the seed-RNNLM using the adaptation data. For more detail of this data, we refer to [35].

[2]For example, "Tokyo-to Chuo-ku" is possible but "Tokyo Chuo-ku" is not possible for [Address 1-2], although both "Tokyo" and "Tokyo-to" are possible for [Address 1].

Table 6 shows the results of changing components for interpolated n-gram LM. #15 – #19 used only baseline data. #20 – #24 used both baseline data and adaptation data. Adding RNNLM generated n-gram components improved WER in the all cases. RNNLM text generation can improve performances even for an English task that has rich data resources. Word RNNLM outperformed subword RNNLM (#16 vs. #18, #17 vs. #19, #21 vs. #23, and #22 vs. #24) for English, which has fewer compound words. Using both forward RNNLM and backward RNNLM had small gain (#16 vs. #17, #21 vs. #23, and #22 vs. # 24), except #18 vs. #19. Performance gain from #20 to #21 – #24 was larger than that from #15 to #16 – #19. This indicates that text generation from RNNLM for an n-gram LM is especially useful for a domain where the amount of training data is not large enough.

5. CONCLUSION

This paper proposed four methods for improving interpolated n-gram LMs using text generated from RNNLM: domain wise generation, subword generation and word acquisition, template generation, and backward generation. All the proposed methods were empirically shown to have positive effects.

These are our recommendations to make an n-gram LM from RNNLM generated text:

- Interpolate n-gram LMs from text generated from multiple RNNLMs for each domain if training data consists of multiple domains. Especially, it is worth interpolating a generated n-gram LM for a domain that has small amounts of data.
- Use subword RNNLM for languages that have many compound words. It is worth acquiring words from text generated from subword RNNLM.
- Use text corpus generated from templates which is generated by RNNLM, if a target domain includes many named entities such as addresses and names.
- Use backward RNNLM to obtain further gain.

6. REFERENCES

[1] George Saon, Gakuto Kurata, Tom Sercu, Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis, Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, Bergul Roomi, and Phil Hall, "English conversational telephone speech recognition by humans and machines," *Proc. Interspeech*, pp. 132–136, 2017.

[2] Wayne Xiong, Lingfeng Wu, Fil Alleva, Jasha Droppo, Xuedong Huang, and Andreas Stolcke, "The Microsoft 2017 conversational speech recognition system," *Proc. ICASSP*, pp. 5934–5938, 2018.

[3] Kyu J Han, Akshay Chandrashekaran, Jungsuk Kim, and Ian Lane, "The CAPIO 2017 conversational speech recognition system," *arXiv preprint arXiv:1801.00059*, 2017.

[4] Haşim Sak, Andrew Senior, Kanishka Rao, and Françoise Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," *Proc. Interspeech*, pp. 1468–1472, 2015.

[5] Rui Wang, Masao Utiyama, Isao Goto, Eiichro Sumita, Hai Zhao, and Bao-Liang Lu, "Converting continuous-space language models into n-gram language models for statistical machine translation," *Proceedings of the 2013 Conference on Em-*

pirical Methods in Natural Language Processing (EMNLP), pp. 845–850, 2013.

[6] Ebru Arisoy, Stanley F Chen, Bhuvana Ramabhadran, and Abhinav Sethy, "Converting neural network language models into back-off language models for efficient decoding in automatic speech recognition," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 22, no. 1, pp. 184–192, 2014.

[7] Heike Adel, Katrin Kirchhoff, Ngoc Thang Vu, Dominic Telaar, and Tanja Schultz, "Comparing approaches to convert recurrent neural networks into backoff language models for efficient decoding," *Proc. Interspeech*, pp. 651–655, 2014.

[8] Ilya Sutskever, James Martens, and Geoffrey E Hinton, "Generating text with recurrent neural networks," *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pp. 1017–1024, 2011.

[9] Ryo Masumura, Taichi Asami, Takanobu Oba, Hirokazu Masataki, Sumitaka Sakauchi, and Akinori Ito, "Combinations of various language model technologies including data expansion and adaptation in spontaneous speech recognition," *Proc. Interspeech*, pp. 463–467, 2015.

[10] Xie Chen, Xunying Liu, Yanmin Qian, MJF Gales, and Philip C Woodland, "CUED-RNNLM – an open-source toolkit for efficient training and evaluation of recurrent neural network language models," *Proc. ICASSP*, pp. 6000–6004, 2016.

[11] Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash, "SRILM at sixteen: Update and outlook," *Proceedings of IEEE automatic speech recognition and understanding workshop (ASRU)*, vol. 5, 2011.

[12] Xie Chen, Tian Tan, Xunying Liu, Pierre Lanchantin, Moquan Wan, Mark JF Gales, and Philip C Woodland, "Recurrent neural network language model adaptation for multi-genre broadcast speech recognition," *Proc. Interspeech*, 2015.

[13] Guangpu Huang, Thiago Fraga da Silva, Lori Lamel, Jean-Luc Gauvain, Arseniy Gorin, Antoine Laurent, Rasa Lileikyte, and Abdel Messouadi, "An investigation into language model data augmentation for low-resourced STT and KWS," *Proc. ICASSP*, pp. 5790–5794, 2017.

[14] Mohamed Hatmi, Christine Jacquin, Emmanuel Morin, and Sylvain Meignier, "Named entity recognition in speech transcripts following an extended taxonomy," *First Workshop on Speech, Language and Audio in Multimedia*, 2013.

[15] Mohamed Ameur Ben Jannet, Olivier Galibert, Martine Adda-Decker, and Sophie Rosset, "Investigating the effect of ASR tuning on named entity recognition," *Proc. Interspeech*, pp. 2486–2490, 2017.

[16] Ziang Xie, "Neural text generation: A practical guide," *arXiv preprint arXiv:1711.09534*, 2017.

[17] Ebru Arisoy, Abhinav Sethy, Bhuvana Ramabhadran, and Stanley Chen, "Bidirectional recurrent neural network language models for automatic speech recognition," *Proc. ICASSP*, pp. 5421–5425, 2015.

[18] Xie Chen, Anton Ragni, Xunying Liu, and Mark Gales, "Investigating bidirectional recurrent neural network language models for speech recognition," *Proc. Interspeech*, 2017.

[19] Gakuto Kurata, Bhuvana Ramabhadran, George Saon, and Abhinav Sethy, "Language modeling with highway LSTM," *Proceedings of IEEE automatic speech recognition and understanding workshop (ASRU)*, pp. 244–251, 2017.

[20] Shaojie Bai, J Zico Kolter, and Vladlen Koltun, "Trellis networks for sequence modeling," *arXiv preprint arXiv:1810.06682*, 2018.

[21] Takaaki Hori, Yotaro Kubo, and Atsushi Nakamura, "Real-time one-pass decoding with recurrent neural network language model for speech recognition," *Proc. ICASSP*, pp. 6364–6368, 2014.

[22] Yinghui Huang, Abhinav Sethy, and Bhuvana Ramabhadran, "Fast neural network language model lookups at n-gram speeds," *Proc. Interspeech*, pp. 274–278, 2017.

[23] Hagen Soltau, Hank Liao, and Hasim Sak, "Neural speech recognizer: Acoustic-to-word LSTM model for large vocabulary speech recognition," *arXiv preprint arXiv:1610.09975*, 2016.

[24] Kartik Audhkhasi, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, and Michael Picheny, "Building competitive direct acoustics-to-word models for english conversational speech recognition," *Proc. ICASSP*, pp. 4759–4763, 2018.

[25] Rukmini Iyer, Mari Ostendorf, and Herbert Gish, "Using out-of-domain data to improve in-domain language models," *IEEE Signal processing letters*, vol. 4, no. 8, pp. 221–223, 1997.

[26] Ryo Masumura, Taichi Asami, Takanobu Oba, Hirokazu Masataki, Sumitaka Sakauchi, and Akinori Ito, "Domain adaptation based on mixture of latent words language models for automatic speech recognition," *IEICE Transactions on Information and Systems*, vol. 101, no. 6, pp. 1581–1590, 2018.

[27] Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky, "Subword language modeling with neural networks," *preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf)*, vol. 8, 2012.

[28] Taku Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," *arXiv preprint arXiv:1804.10959*, 2018.

[29] Oh-Wook Kwon and Jun Park, "Korean large vocabulary continuous speech recognition with morpheme-based recognition units," *Speech Communication*, vol. 39, no. 3-4, pp. 287–300, 2003.

[30] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[31] Stanley F Chen and Joshua Goodman, "An empirical study of smoothing techniques for language modeling," *Computer Speech & Language*, vol. 13, no. 4, pp. 359–394, 1999.

[32] George Saon, Hong-Kwang J Kuo, Steven Rennie, and Michael Picheny, "The IBM 2015 English conversational telephone speech recognition system," *arXiv preprint arXiv:1505.05899*, 2015.

[33] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[34] Tom Sercu, Christian Puhrsch, Brian Kingsbury, and Yann LeCun, "Very deep multilingual convolutional neural networks for lvcsr," in *Proc. ICASSP*, 2016, pp. 4955–4959.

[35] Yinghui Huang, Samuel Thomas, Masayuki Suzuki, Zoltan Tuske, Larry Sansone, and Michael Picheny, "Customizing broadcast news speech recognition systems with closed caption data: a case study," *Submitted to ICASSP*, 2019.