Application-Agnostic Language Modeling for On-Device ASR

Markus Nußbaum-Thom **Lyan Verwimp** **Youssef Oualil**
Apple
{mnussbaumthom,lverwimp,youalil}@apple.com

Abstract

On-device automatic speech recognition systems face several challenges compared to server-based systems. They have to meet stricter constraints in terms of speed, disk size and memory while maintaining the same accuracy. Often they have to serve several applications with different distributions at once, such as communicating with a virtual assistant and speech-to-text. The simplest solution to serve multiple applications is to build application-specific (language) models, but this leads to an increase in memory. Therefore, we explore different data- and architecture-driven language modeling approaches to build a single application-agnostic model. We propose two novel feed-forward architectures that find an optimal trade off between different on-device constraints. In comparison to the application-specific solution, one of our novel approaches reduces the disk size by half, while maintaining speed and accuracy of the original model.

1 Introduction

On-device Automatic Speech Recognition (ASR) is subject to several constraints: it should return accurate results in a reasonable time frame without consuming too much memory and disk space. State-of-the-art research often is accuracy focused, while resource-constrained applications also need to take care of performance and size. Finding an architecture that reaches all constraints is not trivial.

Another challenge is that ASR systems often serve a large variety of requests. ASR systems can serve an on-device Virtual Assistant (VA) but also allow dictated messages, notes, e-mails, etc. – we refer to the latter application as Speech-to-text (STT). Typical VA requests are knowledge-driven questions such as *"how old is Barack Obama?"* or commands, e.g. *"play some Lady Gaga music"*. STT requests are longer and of a different nature than typical VA requests. The solution that yields the best accuracy for both VA and STT is to train separate models for each application, but additional model size is prohibitive. We aim to develop a single model instead.

In this paper, we focus on a Neural Network Language Model (NNLM) in the ASR system. Our baseline is a Fixed-size Ordinally-Forgetting Encoding (FOFE) feed-forward NNLM (Zhang et al., 2015). In ASR, the search space can easily increase so we have to limit the context length used in decoding to reach an acceptable latency and lower memory. Given this short context length, we find that the FOFE feed-forward LM is competitive to the Transformer (Vaswani et al., 2017) in terms of accuracy and better in terms of latency. Irie (2020) has also shown that Transformers are less robust to short context lengths.

To build a single Application-Agnostic (AA) NNLM, we developed a method to optimally sample training data. We sample data from different sources, e.g. anonymized and randomly sampled user requests from opted-in users for VA and STT and artificial requests spanning many different domains that focus on improving the tail of the distribution. The data-driven approach tries to find the optimal balance between the application-specific data sources by creating a balanced development set and distributing the sampling weights based on the importance of each data source and each application on that development set.

Training a single FOFE NNLM on the combined dataset can lead to accuracy degradations, even with a larger model or longer training. We explore two extensions to the baseline FOFE NNLM: firstly, a Mixture FOFE NNLM (Oualil and Klakow, 2017; Irie et al., 2018) which is composed of an ensemble of parallel sub-networks and a mixture sub-network generating normalized probabilities across all sub-networks. These mixture weights are used to compute a weighted average of the ensemble before the softmax output. The second extension is an Application-Dependent (AD) FOFE NNLM

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 5: Industry Track, pages 268–275

that has different sub-networks for each application. At training time, data and gradients are (back-)propagated only through the corresponding sub-network belonging to an application. At inference time, the way the user invokes ASR tells us which application is needed (wake phrase = VA, microphone button = STT) and only the sub-network belonging to the active application is used. Both approaches are able to match or outperform the application-specific model. While the accuracy of the mixture NNLM is slightly better than the AD-NNLM the situation is reversed in terms of speed.

The contributions of this paper are as follows:

- We propose a method to optimally combine application-specific data sources to train an application-agnostic LM in Section 3.
- We propose two novel FOFE-based neural LMs in Section 4 that each match the accuracy of two application-specific language models.
- In Section 6 we compare the novel NNLMs accuracy and speed against the baseline FOFE and state-of-art Transformer models. We do this for three different languages - US English, German and Mandarin Chinese - and three types of test sets (see Section 5 for more information).

2 Related work

We start by discussing related work on modeling several domains/tasks at once. Many pattern recognition tasks are imbalanced since data from different categories do not occur at the same frequency. Therefore, the less frequent categories are not well represented in the training data (Anand et al., 1993; Johnson and Khoshgoftaar, 2019), which results in a sub-optimal model. Data-driven approaches to deal with the data imbalance include under- and over-sampling (Van Hulse et al., 2007). Refinements of these methods select data more intelligently (Kubat and Matwin, 1997; Chawla et al., 2002; Zhang and Mani, 2003; Barandela et al., 2004).

Others approaches modify the training and/or model architecture. Curriculum Learning (Bengio et al., 2009; Shi et al., 2015) emphasizes data by fine-tuning towards the corpus consumed by the end of training. Smith et al. (2020) experiment with multi-task learning, data augmentation and a classifier combined with single-task models to appropriately model several skills in a conversation agent. Balancing through interleaved sampling of different corpora was investigated in (Xing et al., 2022) as well as model-based approaches like multi-task and weighted learning, which allows the model to self-control the impact of different corpora. Other ways to increase the modeling power are using a Mixture of Experts (Shazeer et al., 2017; Zhou et al., 2022) or ensemble networks (Oualil and Klakow, 2017; Irie et al., 2018; Ganaie et al., 2022).

The choice of architecture for language modeling has also been a recurrent topic of research. Early neural LMs use feed-forward layers (Schwenk and Gauvain, 2002; Bengio et al., 2003). Mikolov et al. (2010) introduced recurrent neural LMs that can in principle use unlimited history. These networks are trained with back-propagation through time which 'unrolls' the network in time for gradient computation, but this leads to vanishing gradients (Bengio et al., 1993; Pascanu et al., 2013), essentially limiting the history that can be learned from. Gated recurrent architectures (Sundermeyer et al., 2012; Cho et al., 2014) mitigate this problem.

Recent extensions of the feed-forward architecture have been proposed that alleviate different disadvantages. Zhang et al. (2015) proposed a FOFE, which represents a sequence of words as a vector with fixed length that captures the word order. They show that feed-forward networks with FOFE encoding outperform recurrent models in language modeling. The most widely-used architecture in recent years, is the Transformer (Vaswani et al., 2017) that combines feed-forward layers with multi-head attention, residual connections, and layer normalization (Ba et al., 2016). It has been successfully applied to ASR, see e.g. (Irie et al., 2019; Beck et al., 2020). In this paper, we compare FOFE feed-forward LMs with Transformer LMs and two extensions of the base FOFE feed-forward LMs.

3 Data balancing

The ASR system in this paper serves two applications, VA and STT, for which we observe very different linguistic patterns. To demonstrate these differences, we calculate statistics on two English development sets. Each data set contains 23k anonymized queries and is randomly sampled from real user data similarly to the test sets described in Section 5.

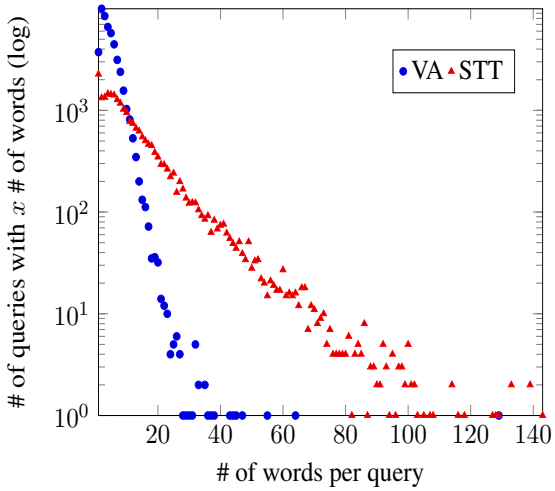


Figure 1: Number of queries (on the y-axis in log scale) with x number of words (on the x-axis) in the English VA and STT dev sets.

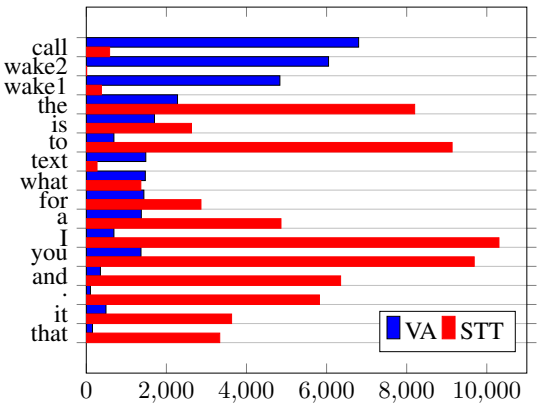


Figure 2: Counts of the union of the 10 most frequent words in both English VA and STT dev sets. "wake2" and "wake1" refer to *"<wakeword_2>* and *"<wakeword_1>*.

VA requests are typically shorter than STT requests. In Figure 1, we plot the number of queries (on a logarithmic scale) that have x number of words for both data sets. For example, in the VA dev set there are 9968 requests with only two words (238 requests consist of only *"<wakeword_1> <wakeword_2>"*), while the STT test set contains 1327 requests with two words. If we define a request of 30 or more words as a 'long' request, we see that the STT test has 2030 long requests while VA has only 21 long requests.

Secondly, the content and style of the requests varies between the two applications. Figure 2 plots the union of the top 10 most frequent words in each data set – ordered by the frequency in the VA dev set. Notice that we allow the user to also dictate punctuation marks, hence the presence of the dot in the list of words. It is clear from this distribution that VA queries are often questions (*what*) or commands (*call, text*) while STT queries are often messages from the perspective of the user (*I, you*) who wants to make their message more readable with punctuation marks.

Because of the different linguistic nature of these two applications, balancing the NNLM training data has a large impact on the quality of the model. A common strategy to determine NNLM sampling weights for each application is to train individual n-gram LMs on each data source and choose relevance weights based on the optimal linear interpolation weights on a development set (Raju et al., 2019). In our setup, the sampling weights for the application-specific text sources are derived from the count merging weights (Bacchiani et al., 2006; Hsu, 2007; Pusateri et al., 2019) instead of a linear combination.

We propose a balancing scheme to derive sampling weights for I text sources that benefit both applications. We create a balanced development set containing approximately the same amount of VA and STT data. Let $\alpha_1, \ldots, \alpha_I \in [0, 1]$ be the sampling weights such that $\sum_{i=1}^{I} \alpha_i = 1$ and $\rho(i) \in \{D, A\}$ indicating if the text source belongs to STT or VA. The redistribution probability masses β_D and β_A for STT and VA respectively are calculated to serve the joint application. These probability masses are determined by the optimal weights that minimize the perplexity of the linear Application-Specific (AS) language model combination on the balanced development set. The application-specific probability mass allocated by each application can be formalized as:

$$\overline{\alpha}_D := \sum_{i,\rho(i)=D} \alpha_i \qquad \text{and} \qquad \overline{\alpha}_A := \sum_{i,\rho(i)=A} \alpha_i.$$

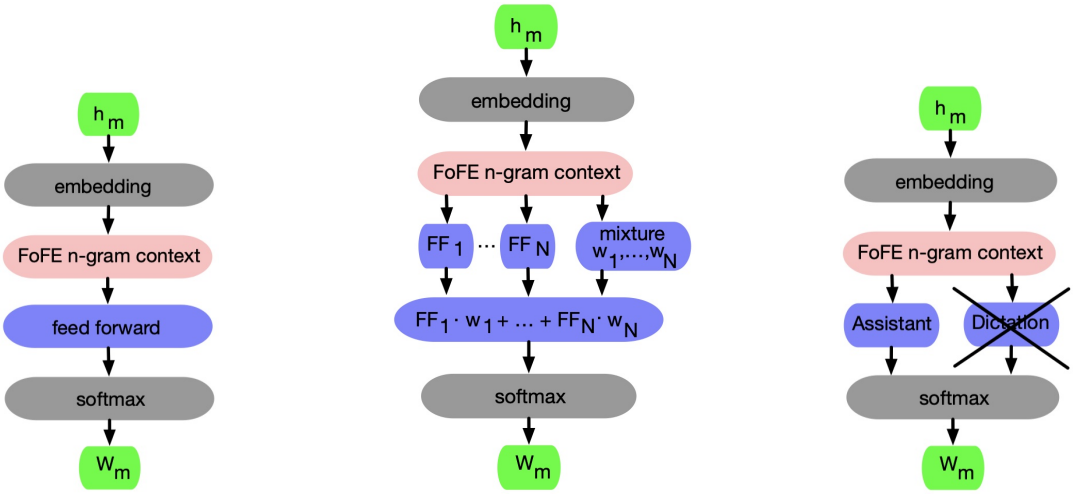
Now consider the ratio between the redistribution and application-specific probability mass:

$$\gamma_A := \frac{\beta_A}{\overline{\alpha}_A} \qquad \text{and} \qquad \gamma_D := \frac{\beta_D}{\overline{\alpha}_D}.$$

These ratios determine the scaling of the original sampling weights to achieve balancing. Balanced sampling weights are then determined by a re-normalization of the scaled sampling weights:

$$\lambda_i := \frac{\gamma_{\rho(i)} \alpha_i}{\sum_j \gamma_{\rho(j)} \alpha_j}, \qquad i = 1, \ldots, I.$$

The heldout and training set for NNLM training is then randomly sampled from the text sources according to the balanced sampling weights.



(a) Base. (b) Mixture (c) Application-dependent.

Figure 3: Let w_m and history h_m be the word and history at position m.

4 Application-Agnostic and Application-Dependent FOFE NNLMs

In this section three different types of NNLM architectures are introduced for on-device ASR. In the following let $w_1^N := w_1, \ldots, w_N$ be a word sequence. All NNLM architectures considered here follow a similar scheme. In each architecture a word embedding is followed by a FOFE layer (Zhang et al., 2015). Let $\alpha > 0$ be the forgetting factor of the FOFE layer and e_m be the word embedding of word w_m then $z_m := z_{m-1} + \alpha \cdot e_m$ generates the FOFE encoding. Afterwards an n-gram context of the FOFE encoding is generated by concatenating n subsequent FOFE encodings for each position: $z_{m-n+1}, \ldots, z_m$. Next, this context is flattened and passed to the hidden layers.

The baseline FOFE NNLM shown in Figure 3a applies a stack of feed-forward layers to the flattened FOFE n-gram context. The output of the last feed-forward layer is fed to a projection layer for dimension reduction before the final softmax layer. This architecture is used for the AS-NNLM, where each application has its own NNLM, as well as for the Application-Agnostic (AA) NNLM, which is trained on balanced data for both applications.

Figure 3b shows the mixture NNLM, which has M parallel sub-networks and a mixture sub-network. Each sub-network is a stack of feed-forward layers. The mixture sub-network is also a stack of feed-forward layers which finish with a softmax output of dimension M to produce mixture weights for each of the parallel sub-networks, similarly to (Oualil and Klakow, 2017; Irie et al., 2018; Zhou et al., 2022) except that the mixture combines FOFE networks. The subsequent layer averages the output of all parallel sub-networks scaled by the corresponding weights of the mixture sub-network softmax output.

Figure 3c shows the Application-Dependent NNLM (AD-NNLM). This architecture uses the application information to train a NNLM in a multi-task style. This NNLM has a separate sub-network and softmax output biases for each application. For training we follow a multi-task approach. The information of the application for each data sample is known and used to select the sub-network and softmax output bias corresponding to the application and only back-propagate through a part of the NNLM. At inference time, data are forwarded through the corresponding sub-network and softmax output bias belonging to the active application.

A word-level NNLM holds the majority of parameters in the embedding. Therefore, the disk size for the mixture and AD-NNLM should increase slightly compared to the baseline architecture. Also the AD-NNLM speed should not increase since it is equivalent to the baseline architecture at inference time.

5 Experimental setup

The training data of our LMs consists of different data sources: anonymized and randomly sampled user requests from both VA and STT that are manually or automatically transcribed, along with synthetic tail-focused datasets. For the latter, we sample from domain-dependent templates and lists

of entities that can fill those slots, both of which are derived from real user data. As mentioned in the introduction, we train NNLMs for three languages: US English, German and Mandarin Chinese.

For our NNLMs, we obtain weights according to the method described in Section 3. For the AS models we sample 6B words while for the AA and AD models we sample 12B words. We run Bayesian hyperparameter optimization and select the final values based on optimal size-accuracy trade off. As a result, the models have a slightly different number of parameters, but we show in section 6 that this does not impact results noticeably. All models have 4 feed-forward layers and an embedding size of 256 – we tie the input and output embedding weights to reduce disk size (Press and Wolf, 2017). The hidden size is 768 for the base FOFE model, 512 for the AD FOFE and mixture FOFE and 256 for the Transformer. The Transformer has the same configuration as Vaswani et al. (2017) and uses 4 attention heads of size 256. We use the top 100k most frequent words as vocabulary. To speed up training, we use Noise Contrastive Estimation (NCE) (Liza and Grzes, 2017) which is replaced by softmax during inference.

We train our NNLMs with Block Momentum Stochastic Gradient Descent (Chen and Huo, 2016) with an initial learning rate of 0.256 for AS, AA and AD FOFE and 1.024 for AA Mixture FOFE. For AS models the optimization converges after 64 epochs while for AA and AD models the optimum is delayed to 128 epochs. We keep the initial learning rate fixed for 16 epochs for AS and 64 epochs for the other models and apply a learning rate decay of 0.7 if the heldout perplexity increases for 4 epochs. To stabilize the training a clip norm of 6.0 is applied and the number of NCE samples is set to 4096.

For evaluation, we test on three types of test sets: (1) VA and (2) STT, which consist of user requests sampled according to the distribution that we observe in our VA/STT and thus contain many head queries, and (3) Tail, which is designed to focus on queries with tail entities. Since these do not occur often in our user data, Tail consists of synthetic requests sampled from the same templates and entity lists that generate the synthetic training data. The requests cover a wide variety of domains such as music, sports and home automation and the audio is generated using Text-to-Speech. Table 1 shows the number of words in each test set.

	VA	STT	Tail
English	226k	292k	454k
German	130k	154k	204k
Mandarin	221k	219k	368k

Table 1: Number of words per test set per language.

We evaluate the accuracy of our models using Word Error Rate (WER) and latency using P95 real-time factor (RTF). If y is the duration of the audio signal and x the time it takes to decode y, RTF is defined as x/y. P95 refers to the 95th percentile and thus captures the latency of the most difficult queries. We run each test three times and average the RTF numbers to capture outliers.

The ASR system uses a deep convolutional neural network acoustic model (AM) as described in (Huang et al., 2020; Pratap et al., 2020). For the AS models, we decode the VA and Tail test sets with a VA-specific NNLM and the STT test sets with a STT-specific NNLM. During decoding, the context length of the NNLMs is limited to 8 words to meet the memory and latency contraints of on-device ASR. We perform a single decoding pass, combining the AM scores with the NNLM scores using optimized weights. We can achieve better WERs by interpolating the NNLM with an n-gram LM trained on tail data and by adding a rescoring pass, but since we want to compare the impact of using different neural architectures, we remove any factors that might obscure that comparison.

6 Results

We first evaluate the accuracy of the different neural architectures. Table 2 reports the WER for different models on the VA, STT and Tail test sets, along with the number of parameters of the model to give an estimate of the size on disk. Note that for the AS FOFE models, we have twice as many parameters as the AA FOFE models because we train two separate models, one for VA+Tail and one for STT.

We first observe that moving from AS to AA FOFE and thus reducing the number of parameters by half gives in some cases 1.5-3.8% WER degradation. Secondly, even though the Transformer architectures have been optimized using Bayesian optimization similar to the FOFE-based models, they give mixed results. For English VA and STT we observe WER improvements while for all other setups we see large degradations.

The AD FOFE model gives the best accuracy

LM	#Par	VA	STT	Tail
English				
AS FOFE	58M	4.02	3.68	**17.48**
AA FOFE	29M	4.11	3.68	17.78
AA Transf.	27M	**3.99**	**3.56**	47.56
AA M-FOFE	37M	**3.99**	**3.56**	17.53
AD FOFE	31M	**3.99**	3.62	17.51
German				
AS FOFE	58M	5.32	6.47	**29.46**
AA FOFE	29M	5.32	6.35	29.93
AA Transf.	27M	11.76	23.34	34.42
AA M-FOFE	37M	5.29	**6.26**	30.37
AD FOFE	31M	**5.25**	6.33	32.36
Mandarin				
AS FOFE	58M	5.17	6.04	39.96
AA FOFE	29M	5.25	6.27	38.84
AA Transf.	27M	8.88	13.29	40.66
AA M-FOFE	37M	5.13	**5.94**	38.16
AD FOFE	31M	**5.12**	6.05	**36.41**
Mandarin (equal number of parameters)				
AS FOFE	68M	5.14	6.00	39.45
AA FOFE	34M	5.26	6.27	38.68
AA Transf.	34M	9.03	13.40	40.38
AA M-FOFE	34M	**5.10**	6.02	38.54
AD FOFE	34M	5.12	**5.98**	**36.48**

Table 2: Number of parameters (#Par) and WERs for the VA, STT and Tail entity test sets for our English, German and Mandarin setups. AS = Application-Specific, AA = Application-Agnostic, AD = Application-Dependent, Transf. = Transformer, M-FOFE = Mixture FOFE.

on VA for all languages, while the AA Mixture FOFE gives the best accuracy on STT, but the differences between the two architectures are small. They outperform the baseline AS/AA FOFE and Transformer models in almost all cases. The only exception are the English and German Tail test sets: the AS FOFE models still achieve the best accuracy, probably because infrequent queries benefit the most from doubling the number of parameters.

As explained in Section 5, we choose hyperparameters based on the optimal accuracy-size trade off. As a result, the number of parameters of the models at the top of Table 2 are not exactly the same. To ensure that the small size differences do not impact the results significantly, we evaluated results for Mandarin models that all have 34M parameters each and added the results at the bottom of Table 2. We observe the same trends: the AD FOFE and AA Mixture FOFE give the best results. We confirm that increasing the number of parameters does not lead to better results.

LM	VA	STT	Tail
English			
AA Transf.	-18.00	-21.75	-11.30
AA M-FOFE	-23.79	-31.54	-17.95
AD FOFE	**7.40**	**-8.04**	**4.66**
German			
AA Transf.	-19.59	-13.92	-24.85
AA M-FOFE	-17.77	-31.45	-79.83
AD FOFE	**7.84**	**3.41**	**5.58**
Mandarin			
AA Transf.	-10.06	-14.04	-8.90
AA M-FOFE	-9.89	-30.21	-36.23
AD FOFE	**-2.11**	**1.63**	**-3.83**

Table 3: Latency results: relative P95 RTF reductions with respect to the AA FOFE models for the VA, STT and Tail entity test sets for our English, German and Mandarin setups. AA = Application-Agnostic, AD = Application-Dependent, Transf. = Transformer, M-FOFE = Mixture FOFE.

Finally, we report the relative change in P95 RTF (P50 RTF showed the same trend) compared to the baseline AA FOFE model in Table 3. Since RTF is hardware-dependent, we mostly care about relative changes compared to the baseline. We observe that both the Transformer and the Mixture FOFE are significantly slower than the baseline. For the English test sets, the Transformer is faster than the Mixture FOFE, while for German and Mandarin speed depends on the test set. The AD FOFE gives the fastest inference speed of the proposed models and even outperforms the vanilla FOFE on English VA and all German test sets, while keeping the degradation limited in the other setups.

7 Conclusion

We aim to develop a single NNLM that can serve both VA and STT requests with the same accuracy and speed as application-specific NNLMs, while reducing the disk size approximately by half. We develop a method to optimally balance the data of the VA and STT applications, and propose two novel FOFE feed-forward architectures. The Application-Agnostic Mixture FOFE and the Application-Dependent FOFE both outperform the baseline FOFE and Transformer models in terms of accuracy, and the latter is also competitive in terms of latency.

Limitations

The two proposed models (AD FOFE and AA FOFE Mixture) have been tested on more languages than the ones mentioned in this paper, but the comparison with Transformer models has not been done for every language. This paper only uses word-level LMs. We have done preliminary experiments with subword-level LMs but more extensive investigation is needed to draw proper conclusions.

Ethics Statement

This paper focuses on the LM of a real-world VA and as such the results cannot be exactly reproduced: we are not aware of any public dataset that mimics our setup, e.g. ASR that can serve both VA and STT applications, training data in several languages that exceeds 6B words along with test sets of several hundreds of thousands of words sampled from real user data, etc. All data have been anonymized and randomly sampled, and human transcription to create the test sets is only performed from opted-in user data.

References

Rangachari Anand, Kishan G. Mehrotra, Chilukuri K. Mohan, and Sanjay Ranka. 1993. An improved algorithm for neural network classification of imbalanced training sets. *IEEE Transactions on Neural Networks*, 4(6):962–969.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. In *arXiv preprint arXiv:1607.06450*.

Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. 2006. MAP adaptation of stochastic grammars. *Computer Speech and Language*, 20:41–68.

Ricardo Barandela, Rosa M. Valdovinos, J. Salvador Sánchez, and Francesc J. Ferri. 2004. The Imbalanced Training Sample Problem: Under or over Sampling? In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 806–814.

Eugen Beck, Ralf Schlüter, and Hermann Ney. 2020. LVCSR with Transformer Language Models. In *Proceedings Interspeech*, pages 1798–1802.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:1137–1155.

Yoshua Bengio, Paolo Frasconi, and Patrice Simard. 1993. The problem of learning long-term dependencies in recurrent networks. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1183–1195.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the Annual International Conference on Machine Learning (ICML)*, volume 382, pages 41–48.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16:321–357.

Kai Chen and Qiang Huo. 2016. Scalable Training of Deep Learning Machines by Incremental Block Training with Intra-block Parallel Optimization and Blockwise Model-Update Filtering. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5880–5884.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

M.A. Ganaie, Minghui Hu, A.K. Malik, M. Tanveer, and P.N. Suganthan. 2022. Ensemble Deep Learning: A Review. *Engineering Applications of Artificial Intelligence*, 115.

Bo-June Hsu. 2007. Generalized linear interpolation of language models. In *IEEE Workshop on Automatic Speech Recognition Understanding (ASRU)*, pages 136–140.

Zhen Huang, Tim Ng, Leo Liu, Henry Mason, Xiaodan Zhuang, and Daben Liu. 2020. SNDCNN: Self-Normalizing Deep CNNs with Scaled Exponential Linear Units for Speech Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6854–6858.

Kazuki Irie. 2020. *Advancing neural language modeling in automatic speech recognition.* Ph.D. thesis, RWTH Aachen University, Germany.

Kazuki Irie, Shankar Kumar, Michael Nirschl, and Hank Liao. 2018. RADMM: Recurrent Adaptive Mixture Model with Applications to Domain Robust Language Modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6079–6083.

Kazuki Irie, Albert Zeyer, Ralf Schlüter, and Hermann Ney. 2019. Language Modeling with Deep Transformers. In *Proceedings Interspeech*, pages 3905–3909.

Justin M. Johnson and Taghi M. Khoshgoftaar. 2019. Survey on deep learning with class imbalance. *Journal of Big Data*, 6.

Miroslav Kubat and Stan Matwin. 1997. Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 179–186.

Farhana Ferdousi Liza and Marek Grzes. 2017. Improving Language Modelling with Noise Contrastive Estimation. In *AAAI Conference on Artificial Intelligence*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings Interspeech*, pages 1045–1048.

Youssef Oualil and Dietrich Klakow. 2017. A neural network approach for mixing language models. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5710–5714.

Razvan Pascanu, Tomáš Mikolov, and Yoshua Bengio. 2013. On the difficulty of training Recurrent Neural Networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1310–1318.

Vineel Pratap, Qiantong Xu, Jacob Kahn, Gilad Avidov, Tatiana Likhomanenko, Awni Hannun, Vitaliy Liptchinsky, Gabriel Synnaeve, and Ronan Collobert. 2020. Scaling Up Online Speech Recognition Using ConvNets. In *Proc. Interspeech 2020*, pages 3376–3380.

Ofir Press and Lior Wolf. 2017. Using the Output Embedding to Improve Language Models. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 157–163.

Ernest Pusateri, Christophe Van Gysel, Rami Botros, Sameer Badaskar, Mirko Hannemann, Youssef Oualil, and Ilya Oparin. 2019. Connecting and comparing language model interpolation techniques. In *Proceedings Interspeech*, pages 3500–3504.

Anirudh Raju, Denis Filimonov, Gautam Tiwari, Guitang Lan, and Ariya Rastrow. 2019. Scalable Multi Corpora Neural Language Models for ASR. In *Proceedings Interspeech 2019*, pages 3910–3914.

Holger Schwenk and Jean-Luc Gauvain. 2002. Connectionist language modeling for large vocabulary continuous speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 765–768.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of- experts layer. In *International Conference on Learning Representations (ICLR)*.

Yangyang Shi, Martha A. Larson, and Catholijn M. Jonker. 2015. Recurrent neural network language model adaptation with curriculum learning. *Computer Speech and Language*, 33(1):136–154.

Eric Michael Smith, Mary Williamson, Kurt Shuster, Jason Weston, and Y-Lan Boureau. 2020. Can You Put it All Together: Evaluating Conversational Agents' Ability to Blend Skills. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2021–2030.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM Neural Networks for Language Modeling. In *Proceedings Interspeech*, pages 194–197.

Jason Van Hulse, Taghi M. Khoshgoftaar, and Amri Napolitano. 2007. Experimental perspectives on learning from imbalanced data. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 935–942.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*.

Yujie Xing, Jinglun Cai, Nils Barlaug, Peng Liu, and Jon Atle Gulla. 2022. Balancing Multi-Domain Corpora Learning for Open Domain Response Generation. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 2104–2120.

Jianping Zhang and Inderjeet Mani. 2003. kNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction. In *Proceedings of the Workshop on Learning from Imbalanced Data Sets*, pages 1–7. ICML.

Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Lirong Dai. 2015. The Fixed-Size Ordinally-Forgetting Encoding Method for Neural Network Language Models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 495–500.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Y. Zhao, Andrew M. Dai, Zhifeng Chen, Quoc Le, and James Laudon. 2022. Mixture-of-experts with expert choice routing. *CoRR*, abs/2202.09368.