RADMM: RECURRENT ADAPTIVE MIXTURE MODEL WITH APPLICATIONS TO DOMAIN ROBUST LANGUAGE MODELING

Kazuki Irie[1]*, Shankar Kumar*[2]*, Michael Nirschl*[2]*, Hank Liao*[2]

[1]Human Language Technology and Pattern Recognition Group
Computer Science Department, RWTH Aachen University, D-52056 Aachen, Germany
[2]Google Inc., New York, NY 10011, USA

irie@cs.rwth-aachen.de,{shankarkumar,mnirschl,hankliao}@google.com

ABSTRACT

We present a new architecture and a training strategy for an adaptive mixture of experts with applications to domain robust language modeling. The proposed model is designed to benefit from the scenario where the training data are available in diverse domains as is the case for YouTube speech recognition. The two core components of our model are an ensemble of parallel long short-term memory (LSTM) expert layers for each domain and another LSTM based network which generates state dependent mixture weights for combining expert LSTM states by linear interpolation. The resulting model is a recurrent adaptive mixture model (RADMM) of domain experts. We train our model on 4.4B words from YouTube speech recognition data. We report results on the YouTube speech recognition test set. Compared with a background LSTM model, we obtain up to 12% relative improvement in perplexity and an improvement in word error rate from 12.3% to 12.1% while using a lattice rescoring with strong pruning.

Index Terms— language modeling, neural networks, speech recognition, mixture of experts, domain adaptation

1. INTRODUCTION

The application of recurrent neural networks for language modeling [1] has been a subject of intensive research this decade. These works cover most of the topics relevant for language modeling in speech recognition, including a number of efforts on the domain adaptation of neural language models [2, 3, 4, 5]. However, when training data in diverse domains are available, no general solution has been investigated in the literature on how to benefit from such a diversity of the data to train a better domain independent neural language model. The standard approach is to train the LSTM language model [6, 7] on the all available data and if the target domain is known, additional fine-tuning for domain adaptation is performed. This contrasts with the case of conventional ngram count models, in which domain specific language models [8, 9] are combined by Bayesian interpolation [10] to build a target domain independent mixture model. In the YouTube speech recognition dataset [11], each video is tagged with one of 17 categories. Motivated by this data diversity, we design a neural network architecture which integrates the diversity of the data into a single neural language model (LM). We present such a model together with a multi-stage training strategy. We evaluate our model on the YouTube speech recognition test set containing various domains, without using any domain information at the evaluation time.

We thank Tom Bagby for his help with the Tensorflow Slim framework. We thank Hasim Sak, Hagen Soltau and Tara Sainath for providing helpful suggestions. We thank Brian Roark for his help with the FST for ngram count models. Kazuki Irie performed the work while interning in the YouTube Speech Group at Google, NY.

2. RECURRENT ADAPTIVE MIXTURE MODEL FOR LANGUAGE MODELING

2.1. Model Description

The architecture of the recurrent adaptive mixture model (RADMM) based language model is shown in Fig. 1. The building blocks of the model are: one word embedding layer shared across experts, multiple layers of parallel LSTM domain *experts*, the *mixer* LSTM network and the single softmax output layer. These components are composed following the equations below which describe the forward pass of the model. The word vector x_t of the input one hot word vector w_t is first obtained by a look up in the *input embedding* matrix W_{emb}:

$$x_t = W_{emb} w_t$$

Such a vector is fed to each domain *expert* LSTM_k for a domain id $k \in 1, .., K$ where K is the number of pre-defined domains,

$$h_t^{(k)}, c_t^{(k)} = \text{LSTM}_k(x_t, h_{t-1}^{(k)}, c_{t-1}^{(k)})$$

where $h_t^{(k)}$ and $c_t^{(k)}$ respectively denote the output and the cell state of the LSTM expert of the domain k.
The same input word vector x_t is also fed to the *mixer* LSTM function:

$$h_t^{(\text{mixer})}, c_t^{(\text{mixer})} = \text{LSTM}_{\text{mixer}}(x_t, h_{t-1}^{(\text{mixer})}, c_{t-1}^{(\text{mixer})})$$

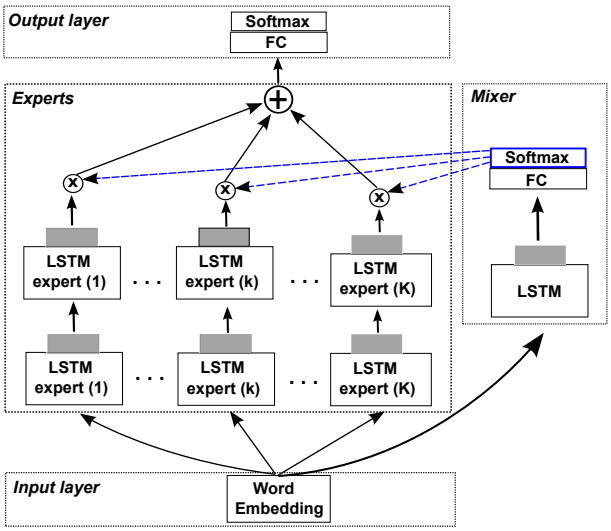


Fig. 1: *Recurrent adaptive mixture model (RADMM) based neural language model.*

which is followed by a fully connected layer with the softmax activation function to generate the mixture weights over K domains:

$$g_t = \text{softmax}(W_{\text{mixer}} h_t^{(\text{mixer})} + b_{\text{mixer}})$$

The scalars $g_t(k)$ are then used as the relevance weights to combine the K LSTM expert features by linear interpolation:

$$s_t = \sum_{k=1}^{K} g_t(k) h_t^{(k)}$$

which is used as the final feature to generate the output word distribution:

$$p(.|w_0^t) = \text{softmax}(W_{\text{out}} s_t + b_{\text{out}})$$

where $w_0^t = w_0, w_1, ..., w_t$ is the word history. We refer to the parameters W_{out} and b_{out} as *output parameters*.

2.2. Training Strategy

2.2.1. Requirements

The role of the mixer is to generate the context dependent relevance weight for each expert. Therefore, training of the mixer requires that the experts are already well trained. Because of this constraint, the training should at least have two stages consisting of pre-training of experts, then training of the mixer. Alternatively, a multi-task approach using the domain prediction loss can be considered to train the mixer: In this work, we train our model only using the language model perplexity as the objective function. In addition, in order to reduce the memory requirement of the model, we tie the input word embedding across different domains (as shown in Fig. 1). Finally, we experimentally found that it is necessary to initialize the final model with the *output parameters* shared across experts to train a good mixer which transfers the performance of the experts to the final model. This requires us to have the shared *input embedding* and the *output parameters* before training the experts, by training a background model beforehand. We therefore end up with a 3-stage training strategy as described in the next sub-section.

Table 1: *YouTube training data split by categories. 'Self Weight' indicates the optimal interpolation weights for 5gram count models trained on each domain when minimizing the perplexity on the subset of the validation set with the same domain (not all domains are in the validation set). 9 categories with the highest self weight are in* ***bold****.*

User Selected Category	Running words	%Total	Self Weight
Autos & Vehicles	31M	0.7%	6%
Comedy	30M	0.7%	29%
Education	758M	17.1%	77%
Entertainment	223M	5.0%	19%
Film & Animation	103M	2.3%	-
Gadgets & Games	79M	1.8%	31%
Howto & Style	149M	3.4%	48%
Movies	409M	9.2%	31%
Music	51M	1.2%	6%
News & Politics	344M	7.8%	27%
Nonprofits & Activism	117M	2.6%	-
People & Blogs	475M	10.7%	31%
Pets & Animals	8M	0.2%	29%
Science & Technology	175M	3.9%	22%
Shows	1.3**B**	29.7%	18%
Sports	61M	1.3%	46%
Trailer	154**K**	0.004%	-
Travel & Events	98M	2.2%	4%
Total	4.4**B**	100%	

2.2.2. 3-stage training recipe

The 3-stage training consists of the following steps. We update or freeze parameters in the 4 blocks (input layer, experts, mixer, and output layer) shown in Fig. 1 at different stages.

1. Train a background LSTM LM using all the data.

2. Take the input embedding and output parameters from the background model to initialize the experts. Keep these parameters constant and train each expert LSTM only using the respective domain data.

3. Take all expert LSTM parameters, input embedding and output parameters from previous stages to initialize the final mixture model. Keep all the experts and input embedding parameters constant and train the mixer LSTM on all the data while fine-tuning the output parameters.

After exploring other strategies, we found that by using this recipe, we can successfully transfer the performance of each expert on their respective domain to the single mixture model. We also include the background model as one of the experts in the mixture model.

3. YOUTUBE SPEECH RECOGNITION EXPERIMENTS

3.1. Dataset

The training data consist of 4.4B running words from around 3.5M YouTube video transcriptions. Each video is tagged with a user selected category. The distribution of the categories in the training data can be found in Table 1. In addition to these training data, we use 71K words from transcriptions of an additional 125 videos as the validation data during the training of neural language models. We evaluate our model on the YouTube evaluation set of 250K words from transcriptions of 296 videos. These data sets are the same as in [12].

3.2. Domain signals in the data

While the second and the third columns of Table 1 shows the diversity of the YouTube data, we can also check whether these user selected categories are relevant for language modeling in the respective category. For this purpose, we train separate 5-gram count models on each domain data. We then compute the interpolation weights that minimize the perplexity on the subset of the validation data corresponding to each category. In the last column of Table 1, the *self weights* indicate the interpolation weight of the domain LM for its own domain. We note that not all domains are present in this validation set. We can observe that the weights are high for most domains, which shows that the definition of domains based on the user selected categories is relevant. From this list, we choose 9 domains which give the greatest self weight, to train the domain experts.

3.3. Neural language model training setups

3.3.1. Basic setups

All neural language models are trained on 32 GPUs using a batch size of 128 and unrolling the recurrence for 20 time steps. We use the Adagrad [13] optimizer with an initial learning rate of 0.2. We use a vocabulary size of 133,008 words. In training, we use the sampled softmax by sampling 4092 words from the Zipf distribution sorted by the unigram frequency. All LSTMs used in this work have tied input and forget gate, as well as the recurrent projection as in Sak et al.'s work [14]. These setups are the same as those used in Kumar et al. [12]. All our implementations of the neural language models are based on the TF-Slim library of Tensorflow [15]. In all models, we use the input word embedding size of 1024. The background model is a 2-layer LSTM with 2048 units per layer with 514 recurrent projection units.

3.3.2. Setups for the RAD mixture model

In the second stage of the training (Sec. 2.2.2), we found that initializing all expert LSTMs with the parameters from the background model is helpful. Therefore, the dimensions of experts are the same as the background LSTM, except in the case of *Education*, where we get slight improvements by increasing the number of units to 4092 and train only on the *Education* data. This is reasonable given the high self weight and the amount of data in this domain shown in Table 1. The same recurrent projection size of 512 is used for all LSTMs. For the sampled softmax, we used the Zipf distribution based on the domain specific unigram frequency to train each expert. The mixer is a 1-layer LSTM with 1024 units and 512 recurrent projection units.

Table 2: *Perplexity overview. The validation perplexities are split by categories. Background and RADMM are* ***single*** *models while Experts are* ***one model per category****.*

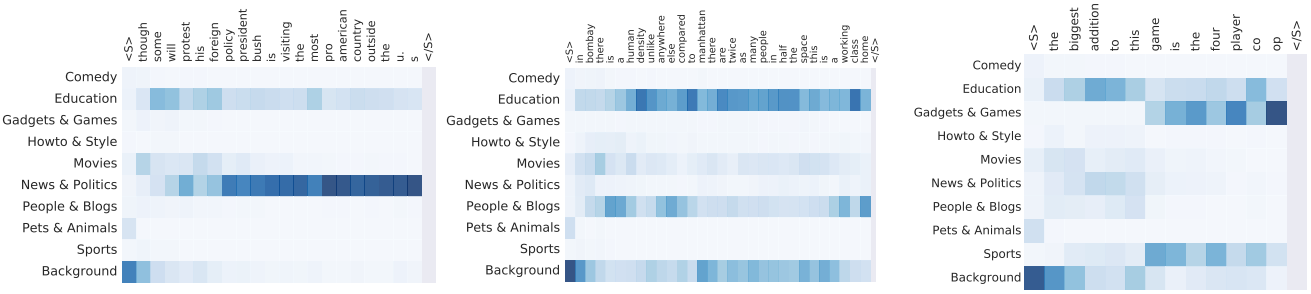
User Selected Category	Background	Experts	RADMM
Comedy	111.3	104.5	107.0
Education	93.7	72.6	78.9
Gadgets & Games	94.9	74.5	86.0
Howto & Style	98.8	81.5	88.6
Movies	145.9	143.4	142.7
News & Politics	155.0	141.6	141.4
People & Blogs	129.0	126.2	121.8
Pets & Animals	98.9	94.5	94.1
Sports	156.0	130.2	140.5
Autos & Vehicles	159.9	-	146.3
Entertainment	139.5	-	132.3
Music	136.8	-	130.7
Science & Technology	112.3	-	104.9
Shows	128.9	-	124.1
Travel & Events	92.3	-	88.4
None	130.9	-	123.9
Full validation set	118.2	-	**109.9**
Evaluation set	61.6	-	**54.0**

3.4. Text based experiments

Table 2 shows the perplexity on the validation set split by the categories. Table 2 has two parts: The upper part shows the perplexities on the domains for which we trained the expert models. We first notice that on some domains such as *Gadgets*, the RADMM does not achieve the performance of the domain specific expert model although it outperforms the background model. However, overall, we can observe that the performance of the different expert models are well transferred to the single RADMM which does not use any explicit domain information at the evaluation time. In addition, the lower part of Table 2 shows that the RADMM also gives better perplexities of up to 9% relative on domains on which we did not train the expert model. Overall on the full validation set and the evaluation set, the improvements in perplexities of respectively 7% and 12% relative are obtained.

3.5. Effectiveness of the mixer output activations

We can check whether the mixer function is making reasonable decisions. Three example sentences from the validation set are shown in Fig. 2: the experts' domain are indicated on the left and input words are shown on the top. The beginning is the same for all cases: since there is no context, the mixer chooses to mainly use the background model. Fig. 2a is a sentence from *News* and the *News* expert is activated. Now if we look at Fig. 2b, the sentence is again from



(a) *Example 1: Category* ***News & Politics****.* (b) *Example 2: Category* ***News & Politics****.* (c) *Example 3: Category* ***Gadgets & Games****.*

Fig. 2: *Examples of mixer output activations (for each input word and each expert category).*

News, though, *People* and *Education* experts are used instead of *News*. This shows some fuzziness of the user selected categories: the domains suggested by the mixer for this sentence are also reasonable. In the example in Fig. 2c, the sentence is clearly from the category *Gadgets&Games*. We observe that the word *game* triggers both *Gadgets* and *Sport* experts, which is also meaningful. This example also shows that the RADMM is robust to domain transitions.

3.6. Lattice rescoring experiments

We apply the neural language model in the second pass lattice rescoring. The lattices are generated by decoding with the first pass 5-gram count model with about 50M ngrams and a vocabulary size of 947K. The phone-level CTC based acoustic model described in [16] is used. We use the push-forward algorithm [17, 18] for lattice rescoring using a strong pruning by keeping only the best hypothesis per node [12]. We only use the second pass LM scores as the linear interpolation with the first pass LM scores did not improve the word error rate (WER). The WERs can be found in Table 3. Despite this strong pruning during the rescoring, the word error rate improves from 12.3% to 12.1%. Given the improvement in perplexity, there is still potential for improvements in WER by improving the search strategy during the rescoring (at the cost of a higher computation time).

Table 3: *WER results on the eval set by lattice rescoring. Perplexities computed on the second pass 133K vocabulary.*

LM	PPL	WER
5-gram count	-	13.0 %
Background LSTM	61.6	12.3 %
RADMM	**54.0**	**12.1** %

3.7. Scaling up

The mixture model has more parameters than the background model since it includes the background model as one of its experts. For the RADMM to have a comparable number of parameters as the background LSTM, we would require each of the experts to have very few parameters, thus, decreasing the modeling capacity. Instead, we investigate how our mixture model can scale up when we increase the size of the background model as well as that of all experts. Table 4 shows the perplexity results of the models with 8192 units in all expert LSTMs. All other dimensions remain the same. In this experiment, we initialize all experts using the background model. We observe that we still get improvements in perplexity of 6% relative on the evaluation set. While simply increasing the LSTM size of a background model has limits[1], we believe that we can get further improvements by increasing the number of experts.

Table 4: *Perplexities of models based on* ***8192****-unit LSTMs.*

LM	Valid	Eval
Background LSTM	105.7	51.0
RADMM	**100.7**	**47.8**

4. RELATED WORK

The RADMM is a new model in the family of mixture of experts proposed in Hampshire and Waibel [19] and Jacobs et al.'s work [20], which was used with recurrent experts in [21]. The mixture of experts has been revisited recently as a general purpose feedforward layer in Shazeer et al.'s work [22]. We focused on building a single domain robust language model in the spirit of the Bayesian interpolation [10] for the ngram count LMs. We achieved this goal by using an adaptive state dependent mixture weights based on the LSTM. This objective differs from previous approaches for using K-component neural LMs [23, 24]. Our model is similar to prior approaches that employ a gating function for combining neural models [25, 26] and domain-experts [27].

5. CONCLUSION

We designed a neural network architecture motivated by data diversity. Our proposed model combines domain adaptation with an LSTM based mixture of experts in a single domain robust model. We developed a training recipe which makes such a fusion possible. We obtained improvements in both perplexity and WER. We observed that the mixer's decisions are meaningful. However, the perplexity of the mixture model was not better than that of experts on some domains: In the future we will work on improving the training strategy of the mixer. Also, the computational cost of the model is high since we run all experts for each prediction. We will investigate a possibility for faster evaluation by making the mixing weights sparser, and first running the mixer before the experts.

[1] We could not achieve a better perplexity by scaling up to 16384 units: the best background perplexity we achieved was 110.0 on the validation set.

6. REFERENCES

[1] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur, "Recurrent neural network based language model," in *Proc. Interspeech*, Makuhari, Japan, Sept. 2010, pp. 1045–1048.

[2] Aram Ter-Sarkisov, Holger Schwenk, Loic Barrault, and Fethi Bougares, "Incremental adaptation strategies for neural network language models," in *Proc. Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, Beijing, China, July 2015, pp. 48–56.

[3] Zoltán Tüske, Kazuki Irie, Ralf Schlüter, and Hermann Ney, "Investigation on log-linear interpolation of multi-domain neural network language model," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Shanghai, China, Mar. 2016.

[4] Siva R. Gangireddy, Pawel Swietojanski, Peter Bell, and Steve Renals, "Unsupervised adaptation of recurrent neural network language models," in *Proc. Interspeech*, San Francisco, CA, USA, Sept. 2016, pp. 2333–2337.

[5] Min Ma, Michael Nirschl, Fadi Biadsy, and Shankar Kumar, "Approaches for neural-network language model adaptation," in *Proc. Interspeech*, Stockholm, Sweden, Aug. 2017, pp. 259–263.

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

[7] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney, "LSTM neural networks for language modeling.," in *Proc. Interspeech*, Portland, OR, USA, Sept. 2012, pp. 194–197.

[8] Reinhard Kneser and Volker Steinbiss, "On the dynamic adaptation of stochastic language models," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Minneapolis, MN, USA, Apr. 1993, pp. 586–589.

[9] Rukmini Iyer and Mari Ostendorf, "Modeling long distance dependence in language: Topic mixtures versus dynamic cache models," *IEEE Trans. Speech and Audio Processing*, vol. 7, no. 1, pp. 30–39, 1999.

[10] Cyril Allauzen and Michael Riley, "Bayesian language model interpolation for mobile speech input," in *Proc. Interspeech*, Florence, Italy, Aug. 2011, pp. 1429–1432.

[11] Hank Liao, Erik McDermott, and Andrew Senior, "Large scale deep neural network acoustic modeling with semi-supervised training data for youtube video transcription," in *Proc. IEEE Automatic Speech Recog. and Understanding Workshop (ASRU)*, Olomouc, Czech Republic, Dec. 2013, pp. 368–373.

[12] Shankar Kumar, Michael Nirschl, Daniel Holtmann-Rice, Hank Liao, Ananda Theertha Suresh, and Felix Yu, "Lattice rescoring strategies for long short-term memory language models in speech recognition," in *Proc. IEEE Automatic Speech Recog. and Understanding Workshop (ASRU)*, Okinawa, Japan, Dec. 2017.

[13] John Duchi, Elad Hazan, and Yoram Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[14] Hasim Sak, Andrew W. Senior, and Françoise Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. Interspeech*, Singapore, Sept. 2014, pp. 338–342.

[15] Martin Abadi et al., "Tensorflow: A system for large-scale machine learning," in *Proc. USENIX Symposium on Operating Systems Design and Implementation*, Savannah, GA, USA, Nov. 2016, pp. 265–283.

[16] Hagen Soltau, Hank Liao, and Hasim Sak, "Neural speech recognizer: Acoustic-to-word lstm model for large vocabulary speech recognition," in *Proc. Interspeech*, Stockholm, Sweden, Aug. 2017, pp. 3707–3711.

[17] Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig, "Joint language and translation modeling with recurrent neural networks," in *Proc. of Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, Seatle, WA, USA, Oct. 2013, pp. 1044–1054.

[18] Martin Sundermeyer, Zoltán Tüske, Ralf Schlüter, and Hermann Ney, "Lattice decoding and rescoring with long-span neural network language models," in *Proc. Interspeech*, Singapore, Sept. 2014, pp. 661–665.

[19] John B Hampshire and Alex Waibel, "The meta-pi network: Building distributed knowledge representations for robust multisource pattern recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 7, pp. 751–769, 1992.

[20] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton, "Adaptive mixtures of local experts," *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.

[21] Jun Tani and Stefano Nolfi, "Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor systems," *Neural Networks*, vol. 12, no. 7, pp. 1131–1141, 1999.

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

[23] Yangyang Shi, Martha Larson, Pascal Wiggers, and Catholijn M. Jonker, "K-component adaptive recurrent neural network language models," in *Proc. Int. Conf. on Text, Speech, and Dialogue (TSD)*, Pilsen, Czech Republic, Sept. 2013, pp. 311–318.

[24] Youssef Oualil and Dietrich Klakow, "A neural network approach for mixing language models," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, LA, USA, Mar. 2017, pp. 5710–5714.

[25] Ekaterina Garmash and Christof Monz, "Ensemble learning for multi-source neural machine translation," in *Proc. Int. Conf. on Comp. Linguistics (COLING)*, Osaka, Japan, Dec. 2016, pp. 1409–1418.

[26] Jian Zhang, Xiaofeng Wu, Andy Way, and Qun Liu, "Fast gated neural domain adaptation: Language model as a case study," in *Proc. Int. Conf. on Comp. Linguistics (COLING)*, Osaka, Japan, Dec. 2016, pp. 1386–1397.

[27] Young-Bum Kim, Karl Stratos, and Dongchan Kim, "Domain attention with an ensemble of experts," in *Proc. Association for Computational Linguistics (ACL)*, Vancouver, Canada, July 2017, pp. 643–653.