NEURAL-FST CLASS LANGUAGE MODEL FOR END-TO-END SPEECH RECOGNITION

Antoine Bruguier[*], *Duc Le*[*], *Rohit Prabhavalkar*[†], *Dangna Li, Zhe Liu, Bo Wang, Eun Chang, Fuchun Peng, Ozlem Kalinli, Michael L. Seltzer*

Facebook AI, USA
tonybruguier@fb.com

ABSTRACT

We propose Neural-FST Class Language Model (NFCLM) for end-to-end speech recognition, a novel method that combines neural network language models (NNLMs) and finite state transducers (FSTs) in a mathematically consistent framework. Our method utilizes a background NNLM which models generic background text together with a collection of domain-specific entities modeled as individual FSTs. Each output token is generated by a mixture of these components; the mixture weights are estimated with a separately trained neural decider. We show that NFCLM significantly outperforms NNLM by 15.8% relative in terms of Word Error Rate. NFCLM achieves similar performance as traditional NNLM and FST shallow fusion while being less prone to overbiasing and 12 times more compact, making it more suitable for on-device usage.

Index Terms— class language model, shallow fusion, end-to-end speech recognition, named entities

1. INTRODUCTION

End-to-end (E2E) automatic speech recognition (ASR) models are becoming increasingly popular, especially for on-device applications, due to their compact size and competitive transcription accuracy [1,2]. Unlike conventional hybrid ASR systems [3], which consist of separately trained acoustic (AM) and language models (LM), E2E models are trained jointly on paired acoustic-text data. If paired data are available for all domains of interest, then E2E models can be made more robust by training on diverse data [4]. However, in many practical situations it is unreasonable to expect that paired data are available for all domains, either because such data collection would be prohibitively expensive or impractical. In order to ensure that E2E ASR systems can be suitable replacements for conventional hybrid systems, we are presented with two challenges: rapidly adapting the E2E ASR system to support a new domain of interest, for which only text data might be available; and, ensuring that E2E ASR models can correctly recognize rare words (typically named entities), which might be unseen during training. The first of these challenges has been addressed in previous work through *language model fusion* [5–7]. The second challenge, recognizing rare words, has only been most successfully addressed in the setting where we have a set of entities (e.g., user's contacts, song playlists, etc...) through *contextual biasing* [8–10].

For this paper, we focus on the problem of developing an E2E ASR model which can perform well on multiple domains, where a subset of domains are only available as a set of context-free grammar (CFG) rules [11]. This setting presents an important challenge when building speech-enabled voice assistants that must be able to support downstream spoken language understanding (SLU) systems which constantly expand coverage by serving new domains. One approach to this problem is to generate sentences from the underlying domain grammar, and then generate the corresponding acoustic sequences using a text-to-speech (TTS) system (e.g., [1, 12, 13]). However, this approach is computationally expensive, both the TTS synthesis process as well as the effort required to re-train the E2E model, which makes it harder to rapidly adapt to a new domain. In this work we focus on using a fixed pre-trained E2E ASR model and improving the LM used during decoding.

The simplest approach to building neural LMs which cover multiple domains, some of which are represented as CFGs, is to generate domain-specific data by substituting non-terminal symbols [14, 15]. This data can then be pooled together with the "background" text to train a single neural LM with wider coverage. While this solution (which serves as one of our baselines) is simple to implement, it requires model re-training to ingest new patterns or entities. Extending this solution to work without re-training is not trivial for *neural LMs*; some early work along this direction was explored in [10]. An alternative approach, employed extensively in hybrid ASR systems [3] with *n-gram LMs*, utilizes class-based non-terminals in the decoder graph to represent classes of interest (e.g., contact names, songs, artists, etc...); these can then be replaced on-the-fly using separate finite state transducers (FSTs) [16] which can be compiled and personalized for each user [17–19]. The corresponding solution for E2E models typically involves shallow fusion contextual biasing [8–10, 20]. The FST-based biasing technique has two disadvantages: it is somewhat challenging to set the weights in the biasing FST, which are typically hand-tuned on a development set; and, there is a danger of *overbiasing* phrases if the biasing weights are set to inappropriate values, although this can be mitigated somewhat by only allowing the biasing to be applied in specific contexts [1]. One possibility is to use the ASR model's output to decide when to apply biasing [21] but this means that the LM is tied to the ASR model.

In this work, we propose Neural-FST Class Language Model (NFCLM), an approach that builds a factorized LM consisting of a *background* neural LM [22, 23], which is intended to have broad coverage for general text data, along with several separate domain-specific LMs. Each word in the sentence is generated by this mixture of LM components weighted by a separately estimated neural *decider*. While our work is similar to previous work on mixture-of-expert LMs [24, 25], a crucial aspect of our proposed model is that the domain specific LMs are represented using FSTs, analogous to their use in shallow fusion biasing [8–10]. However, in our work, these are integrated in a purely probabilistic model without the need for hand-tuning of biasing weights. Leveraging FSTs to represent the domain-specific entities makes it easier to ingest new

[*]Equal contribution. [†]Work performed while at Facebook. We'd like to thank Jun Liu and Frank Seide for their helpful comments and suggestions.

entities into the LM, which allows for rapid adaptation to a new domain or rare named entities. The proposed NFCLM is used in shallow fusion with internal LM subtraction [26] during decoding. Our experiments demonstrate that NFCLM outperforms NNLM in modeling rare named entities, reducing Word Error Rate (WER) by 15.8% relative on an entity-heavy evaluation set. Compared to the traditional NNLM and FST shallow fusion, NFCLM achieves similar WER while being less prone to overbiasing and 12 times more compact, making it a more suitable choice for on-device ASR.

2. NEURAL-FST CLASS LANGUAGE MODEL (NFCLM)

2.1. Theoretical Formulation

Let us assume we have an end-to-end ASR model trained to output symbols over some sub-word vocabulary, $\mathcal{V}$ (WordPieces [27], in this work). We introduce another set $\mathcal{C}$ consisting of non-terminal background (`@bg`) and non-background (e.g., `@song`, `@artist`) classes, which is disjoint from $\mathcal{V}$. For convenience, we define a special symbol ϵ to represent the continuation of an existing non-background class, and an expanded set $\mathcal{C}_\epsilon = \mathcal{C} \cup \{\epsilon\}$.

Given a length-n output symbol history $\mathbf{h^w} \in \mathcal{V}^n$, let $\mathbf{h^c} \in \mathcal{C}_\epsilon^n$ be a possible class alignment of $\mathbf{h^w}$, which indicates that the symbol h_i^w is generated by the class h_i^c. Then, we can define the NFCLM probability for the next output symbol w by marginalizing over all possible alignments and all possible classes c which can generate w:

$$P(w|\mathbf{h^w}) = \sum_{\mathbf{h^c} \in \mathcal{C}_\epsilon^n} P(\mathbf{h^c}|\mathbf{h^w}) \sum_{c \in \mathcal{C}_\epsilon} P(w|c, \mathbf{h^c}, \mathbf{h^w}) P(c|\mathbf{h^c}, \mathbf{h^w}) \tag{1}$$

As can be seen in Eq. (1), the overall NFCLM consists of the *class component probability* $P(w|c, \mathbf{h^c}, \mathbf{h^w})$, the *class emission probability* $P(c|\mathbf{h^c}, \mathbf{h^w})$, and the *alignment probability* $P(\mathbf{h^c}|\mathbf{h^w})$. Note that unlike previous work on class-based LMs [28], we do not assume that each output symbol belongs to a fixed class in $\mathcal{C}_\epsilon$; this is especially important since we model sub-word units which by their nature are shared across all classes.

2.1.1. *Class Component Probability: $P(w|c, \mathbf{h^c}, \mathbf{h^w})$*

The goal of this distribution is to estimate the probability of producing the output symbol w, given the output symbol history $\mathbf{h^w}$, its alignment $\mathbf{h^c}$, and the assumption that w is generated by the class c. Let $f(\mathbf{h^c}, c)$ be a function that returns the last non-epsilon class in the sequence $\{\mathbf{h^c}, c\}$ if there is any (i.e., the class we are currently in), or ϵ otherwise (i.e., no class has been emitted yet). Then we can formally define the class component probability:

$$P(w|c, \mathbf{h^c}, \mathbf{h^w}) = \begin{cases} P_{\text{@bg}}(w|\mathbf{h^w}) & \text{if } f(\mathbf{h^c}, c) = @bg \\ P_{f(\mathbf{h^c},c)}(w|\mathbf{h^c}, \mathbf{h^w}) & \text{else if } f(\mathbf{h^c}, c) \neq \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

As can be seen in Eq. (2), we model the background and non-background class LMs differently. The background class LM, $P_{\text{@bg}}(w|\mathbf{h^w})$, models generic open-domain text, thus it does not take $\mathbf{h^c}$ into account and can be viewed as a standard neural LM over $\mathcal{V}$. For non-background classes, our goal is to allow easy injection of new entities into the model without re-training, thus we represent them as FSTs built from class-specific data. Unlike the FST in [8] which used hand-tuned biasing weights, our FST is constructed so that the weights can be interpreted as probabilities, i.e., the weights of all outgoing arcs from any given state add up to 1.0. More formally, these FSTs estimate $P_c(w|\mathbf{h^c}, \mathbf{h^w}) = P_c(w|\mathbf{h_c^{\text{prefix}}})$, where $\mathbf{h_c^{\text{prefix}}}$ is defined as the last symbols in $\mathbf{h^w}$ that are generated by the non-background class c or its continuation. For example, given $\mathbf{h^w} = \{\_play, \_ro, sie\}$ and $\mathbf{h^c} = \{@bg, @song, \epsilon\}$, then $\mathbf{h_{\text{@song}}^{\text{prefix}}} = \{\_ro, sie\}$ and $\mathbf{h_{\text{@artist}}^{\text{prefix}}} = \{\}$.

Let us also define $P_c(\phi|\mathbf{h_c^{\text{prefix}}})$ as the probability of exiting from the FST of the non-background class c after consuming $\mathbf{h_c^{\text{prefix}}}$. This probability will be zero if we end up in a non-final FST state after consuming $\mathbf{h_c^{\text{prefix}}}$ and non-zero otherwise. For convenience, let $P_{\text{@bg}}(\phi|\mathbf{h^w}) = 1$ so that $P(\phi|c, \mathbf{h^c}, \mathbf{h^w})$ is defined for both the background and non-background classes.

2.1.2. *Class Emission Probability: $P(c|\mathbf{h^c}, \mathbf{h^w})$*

The goal of this distribution is to predict which class c is likely to come next given a history of output symbols $\mathbf{h^w}$ and its alignment $\mathbf{h^c}$. This is computed primarily with a *decider* neural network trained separately to estimate $P_D(c|\mathbf{h^c}, \mathbf{h^w}) = P_D(c|\mathbf{h}^{\text{decider}})$, where $c \in \mathcal{C}$ and the decider history $\mathbf{h}^{\text{decider}}$ is the same as $\mathbf{h^c}$, but with all ϵ removed and $@bg$ tokens replaced with the corresponding WordPiece. For example, if $\mathbf{h^w} = \{\_play, \_ro, sie\}$ and $\mathbf{h^c} = \{@bg, @song, \epsilon\}$, then $\mathbf{h}^{\text{decider}} = \{\_play, @song\}$. By formulating the decider this way, we are able to condition on both regular output symbols as well as the class labels. We can then formally define the class emission probability:

$$P(c|\mathbf{h^c}, \mathbf{h^w}) = \begin{cases} 1 - P(\phi|c, \mathbf{h^c}, \mathbf{h^w}) & \text{if } c = \epsilon \\ P(\phi|c, \mathbf{h^c}, \mathbf{h^w}) P_D(c|\mathbf{h^c}, \mathbf{h^w}) & \text{otherwise} \end{cases} \tag{3}$$

From Eq. (3) and the exit probability's definition in Section 2.1.1, we see that the probability of emitting ϵ is always 0 unless we are in a non-final state of a non-background class FST.

2.1.3. *Alignment Probability: $P(\mathbf{h^c}|\mathbf{h^w})$*

The goal of this distribution is to estimate the probability that the output symbol sequence $\mathbf{h^w}$ corresponds to the alignment $\mathbf{h^c}$. Let us define $\mathbf{h^w_{:k}}$ and $\mathbf{h^c_{:k}}$ as the sub-sequences containing the first k elements in $\mathbf{h^w}$ and $\mathbf{h^c}$, respectively. Then, the alignment probability can be expanded as follows using Bayes' rule:

$$P(\mathbf{h^c}|\mathbf{h^w}) = P(h_1^c, h_2^c, \ldots, h_n^c | h_1^w, h_2^w, \ldots, h_n^w) \tag{4}$$
$$= P(h_1^c) P(h_2^c|\mathbf{h^c_{:1}}, \mathbf{h^w_{:1}}) \ldots P(h_n^c|\mathbf{h^c_{:n-1}}, \mathbf{h^w_{:n-1}}) \tag{5}$$

As shown in Eq. (5), the alignment probability can be computed from the class emission probability of each label in the sequence, using Eq. (3). In practice, we make use of memoization to avoid recomputing this probability each time. Note that $P(h_1^c)$ corresponds to the class emission probability of the first symbol, when both $\mathbf{h^c}$ and $\mathbf{h^w}$ are empty: $P(c) = P(c|\mathbf{h^c} = \{\}, \mathbf{h^w} = \{\})$.

2.2. Practical Implementation

2.2.1. *Beam Search Approximation*

The number of possible alignment paths $\mathbf{h^c}$ grows exponentially with the length of the output symbol history $\mathbf{h^w}$, thus the computation of Eq. (1) is intractable. As with many ASR decoding algorithms, we can approximate this equation with beam search:

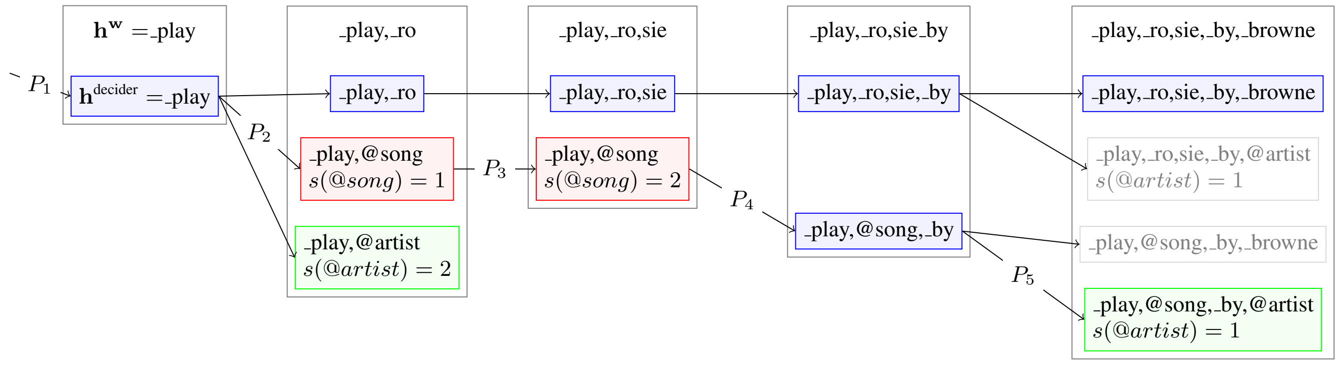

Fig. 1: NFCLM's expanded dynamic FST after consuming the symbol sequence $\mathbf{h}^{\mathbf{w}} = \{\_play, \_ro, sie, \_by, \_browne\}$. The decider histories $\mathbf{h}^{\text{decider}}$ left inside the beam after each extension are displayed on the state label. Here we assume that there are two classes, $@song$ (red) and $@artist$ (green), see Fig. 2, in addition to the background states (blue). In the last extension, two paths are dropped from the beam (gray). As an illustration, $P_1 = P_D(@bg|\{\})P_{@bg}(\_play|\{\})$, $P_2 = P_D(@song|\{\_play\})P_{@song}(\_ro|\{\})$, $P_3 = P_{@song}(sie|\{\_ro\})$, $P_4 = P_D(@bg|\{\_play, @song\})P_{@bg}(\_by|\{\_play, \_ro, sie\})$, and $P_5 = P_D(@artist|\{\_play, @song, \_by\})P_{@artist}(\_browne|\{\})$.

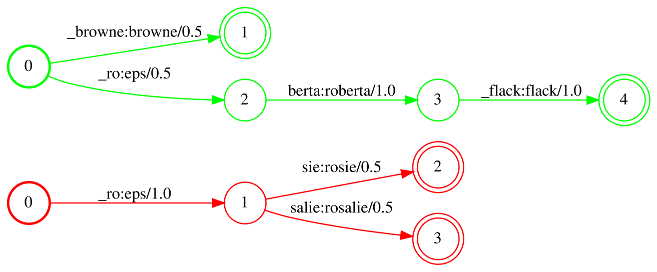

Fig. 2: Component FST for class $@song$ (red) and $@artist$ (green). Note that there is no loop-back from the final states to the start states. The handling of multiple entries into the same class is controlled by the class emission probability, more specifically the neural decider.

$$P(w|\mathbf{h}^{\mathbf{w}}) \approx \sum_{\mathbf{h}^{\mathbf{c}} \in \mathcal{B}_{\mathbf{h}^{\mathbf{w}}}} P(\mathbf{h}^{\mathbf{c}}|\mathbf{h}^{\mathbf{w}}) \sum_{c \in \mathcal{C}_{\epsilon}} P(w|c, \mathbf{h}^{\mathbf{c}}, \mathbf{h}^{\mathbf{w}}) P(c|\mathbf{h}^{\mathbf{c}}, \mathbf{h}^{\mathbf{w}}) \tag{6}$$

where the beam $\mathcal{B}_{\mathbf{h}^{\mathbf{w}}}$ contains the most probable alignments of $\mathbf{h}^{\mathbf{w}}$. In this work, we implement $\mathcal{B}_{\mathbf{h}^{\mathbf{w}}}$ as a soft beam that retains up to $N = 100$ alignments whose log-probability difference compared to the best alignment in the beam is at most $\delta = 30$. Through this approximation, we can now compute $P(w|\mathbf{h}^{\mathbf{w}})$ efficiently.

2.2.2. *Dynamic FST*

Similar to a conventional sequence-to-sequence neural network [29], we can represent NFCLM as an infinite FST where new states and arcs are added dynamically as we extend $\mathbf{h}^{\mathbf{w}}$ with new symbols. This implementation allows NFCLM to be used like a normal FST while wrapping the decider, the background neural LM, and multiple non-background component FSTs under the hood. As an illustration, we show how NFCLM consumes the sequence $\{\_play, \_ro, sie, \_by, \_browne\}$ in Fig. 1. This toy example assumes that we have two classes, $@artist$ and $@song$, with entities $\{(\_ro, berta, \_flack), (\_browne)\}$ and $\{(\_ro, sie), (\_ro, salie)\}$, respectively (Fig. 2). At every decoding step, new class alignments are added and pruned away based on the beam setting.

3. EXPERIMENTAL SETUP

3.1. Text Datasets and Evaluation Procedure

On the training side, we have access to a 29M-sentence (147M-word) *background* text corpus sampled from crowdsourced data and the manually transcribed Facebook voice assistant traffic of users who have agreed to having their voice activity reviewed and analyzed. We also have access to a set of CFG rules that represent the music and weather domains. The rules consist of 291 patterns and seven non-terminals, $@album$ (11.8K entities), $@artist$ (10.9K), $@genre$ (1.0K), $@playlist$ (3.8K), $@song$ (89.0K), $@station$ (5.2K), and $@location$ (48.1K). From these data we construct a 27M-sentence (183M-word) *non-background* text corpus by fully expanding the CFG rules with uniformly sampled patterns and entities. Together, these comprise all the available text data for training.

We consider two evaluation sets in this work. The first one, *General* (GEN), consists of 16.0K manually transcribed utterances collected from the voice activity of users who interacted with the Facebook voice assistant and have agreed to having their voice activity reviewed and analyzed. The second evaluation set, *Entity-Heavy* (ENT), contains 10.0K utterances generated with an in-house TTS engine; each utterance contains on average 3.4 regular words and 3.1 entity words. The references are generated randomly by expanding the CFG rules as described above. We use TTS data for evaluation in this work in order to test the proposed method on more diverse patterns and entities compared to those seen in traffic. Note that using TTS data does not change the overall conclusion, only its magnitude. The goal of external LM fusion is to improve WER on ENT while minimizing degradation on GEN, by leveraging all available text data without modifications to the base ASR model.

3.2. Baseline Systems

Our base E2E ASR model is a sequence transducer (RNN-T) [30] containing approximately 100M parameters; the overall architecture and training methods are described in detail in [10]. To summarize, the model employs a 28-layer streamable low-latency Emformer encoder [31] with a stride of four, 40ms lookahead, and 120ms segment size. The target units are 4095 unigram WordPieces [32] built with SentencePiece [27]. The model is trained on 1.7M hours of in-house data using AR-RNNT loss [33] and trie-based deep biasing [10].

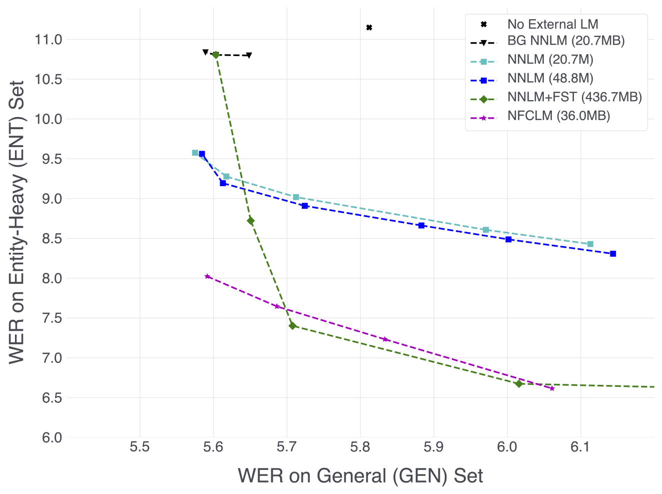

Fig. 3: WER comparison of different LM integration methods on the *General* (GEN) and *Entity-Heavy* (ENT) evaluation sets.

We consider two baseline methods to utilize the text data, both based on 1st-pass shallow fusion. The first method (NNLM) encodes all available text data (background, 147M words, and non-background, 183M words, fed in random order) with a single neural LM consisting of three LSTM layers (20.7M parameters), trained using Cross Entropy (CE) loss and 0.1 dropout. Training sentences are tokenized into WordPieces using the same vocabulary as that of the base ASR model. The second method (NNLM+FST) combines a similar neural LM trained *only on the background text* with a generic biasing FST built on the CFG data. This FST is constructed from the CFG patterns and the non-terminals are statically replaced with class-specific FSTs built on the entity lists (e.g., Fig. 2), following [10]; the final FST is 416MB in size. With this method, we do not need to re-train the neural LM when new entities are added. Note that unlike NFCLM, NNLM+FST treats neural LM and FST as separate systems rather than as components in a unified framework.

3.3. NFCLM Setup

As described in Section 2, NFCLM is defined by the background LM, class-component FSTs, and the decider. We reuse the background neural LM and class-specific FSTs in NNLM+FST for NFCLM to ensure fair comparison. The decider is a small sequence classification model consisting of a single LSTM layer (286K parameters) trained with CE loss and 0.1 dropout. The training set contains a mixture of background (10%) and CFG (90%) data. This strategy exposes the decider to more training data and avoids overfitting to fixed patterns. To re-balance the probabilities produced by the decider, we re-normalize them using the frequency of each output class appearance in the CFG training data; we found that this re-normalization is necessary to prevent the probability mass from being concentrated on the @*bg* class. Since we do not need to perform static FST replacement for NFCLM, the total size taken up by the class-component FSTs is only 15MB, bringing the total size to 36MB (the neural LMs are 8-bit quantized).

4. RESULTS AND DISCUSSION

We assess the performance of each LM integration method through its effect on both the GEN and ENT evaluation sets. We first obtain different operating points for each method by sweeping on various hyperparameters, including neural LM weight, internal LM weight, and FST shallow fusion weight (only applicable to NNLM+FST), which control the WER tradeoff between the two sets. We then plot the convex hull corresponding to each collection of operating points in Fig. 3. We are thus able to visualize and compare the efficient frontiers of different methods; the closer the points are to the origin, the better the method. Note that the vanilla RNN-T result without any external LM is displayed as a single point in the graph, as we cannot control the tradeoff between GEN and ENT in this case.

We can see the benefit of neural LM shallow fusion in Fig. 3 (NNLM), especially on the ENT evaluation set with 15–25% relative WER reduction (WERR) compared to the vanilla RNN-T baseline. A similar neural LM trained with only the background text (BG NNLM) offers relatively modest improvement on ENT, pointing to the importance of the non-background training text. Increasing the model size from 20.7M to 48.8M parameters does not improve the results significantly. Compared to NNLM, FST can better memorize rare named entities. This is confirmed through NNLM+FST achieving significantly better results on ENT; however, the method incurs greater degradation on GEN due to overbiasing.

The proposed NFCLM clearly outperforms NNLM, achieving 15.8% WERR on ENT and similar performance on GEN. The improvement here, similar to NNLM+FST, can be attributed to the use of FSTs for class-specific entities. Compared to NNLM+FST, NFCLM is significantly less prone to overfitting. If we want to obtain the best performance on ENT while minimizing the degradation on GEN (leftmost point of each curve), then NFCLM is the clear winner. Looking at the rest of the curve, NFCLM and NNLM+FST achieve very similar operating points; however, NFCLM uses drastically less storage space (36.0MB vs. 436.7MB, 12 times smaller), making it more amenable to be used on-device. The decoding speed, measured in Real-Time Factor (RTF), with NFCLM is slightly worse (0.28 vs. 0.26) due to the extra computation incurred by the multiple class alignments. By factoring out the model into smaller components, we can handle changes in the entity lists and expand to new domains more quickly. When any class-specific entity list changes, we only need to rebuild the FST for that class. Supporting new domains and patterns does require re-training the decider neural network; however, this is considerably cheaper than re-training the entire NNLM since the decider is much more lightweight and does not require as much data to converge. Moreover, by implementing NFCLM as a dynamic FST, no change to the decoder is needed.

5. CONCLUSION AND FUTURE WORK

In this paper, we proposed NFCLM, a novel method that factorizes the LM into separate components (neural LM and FSTs) in a unified and mathematically consistent framework. We demonstrated that the proposed method, when used in end-to-end ASR, significantly outperforms vanilla NNLM in modeling rare named entities, and is less prone to overbiasing and much more compact than the traditional FST shallow fusion approach.

For future work, we plan on carrying out more in-depth studies to better understand how robust NFCLM is to more varied use cases in terms of entity types, entity test set sizes, and train/test data sets. We also plan to expand the data on which the decider is trained. Instead of focusing on a limited set of manually annotated patterns, we plan to use a large corpus of text and use an entity tagger to construct patterns and entity lists automatically. We will also explore alternative LM architectures beyond LSTMs.

6. REFERENCES

[1] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K. C. Sim, T. Bagby, S.-Y. Chang, K. Rao, and A Gruenstein, "Streaming end-to-end speech recognition for mobile devices," in *Proc. of ICASSP*, 2019.

[2] K. Kim, K. Lee, D. Gowda, J. Park, S. Kim, S. Jin, Y.-Y. Lee, J. Yeo, D. Kim, S. Jung, J. Lee, M. Han, and C. Kim, "Attention based on-device streaming speech recognition with large speech corpus," in *Proc. of ASRU*, 2019.

[3] N. Morgan and H. Bourlard, "Continuous speech recognition," *IEEE Signal Processing Magazine*, vol. 12, no. 3, pp. 24–42, 1995.

[4] A. Narayanan, R. Prabhavalkar, C.-C. Chiu, D. Rybach, T. N. Sainath, and T. Strohman, "Recognizing long-form speech using streaming end-to-end models," in *Proc. of ASRU*, 2019.

[5] C. Gulcehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.-C. Lin, F. Bougares, H. Schwenk, and Y. Bengio, "On using monolingual corpora in neural machine translation," *arXiv preprint arXiv:1503.03535*, 2015.

[6] A. Kannan, Y. Wu, P. Nguyen, T. N. Sainath, Z. Chen, and R. Prabhavalkar, "An analysis of incorporating an external language model into a sequence-to-sequence model," in *Proc. of ICASSP*, 2018.

[7] A. Sriram, H. Jun, S. Satheesh, and A. Coates, "Cold fusion: Training seq2seq models together with language models," in *Proc. of Interspeech*, 2018.

[8] D. Zhao, T. N. Sainath, D. Rybach, P. Rondon, D. Bhatia, B. Li, and R. Pang, "Shallow-fusion end-to-end contextual biasing.," in *Proc. of Interspeech*, 2019.

[9] D. Le, G. Keren, J. Chan, J. Mahadeokar, C. Fuegen, and M. L. Seltzer, "Deep Shallow Fusion for RNN-T Personalization," in *Proc. SLT*, 2021.

[10] D. Le, M. Jain, G. Keren, S. Kim, Y. Shi, J. Mahadeokar, J. Chan, Y. Shangguan, C. Fuegen, O. Kalinli, Y. Saraf, and M. L. Seltzer, "Contextualized Streaming End-to-End Speech Recognition with Trie-Based Deep Biasing and Shallow Fusion," in *Proc. Interspeech*, 2021, pp. 1772–1776.

[11] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to automata theory, languages, and computation*, ACM, 2001.

[12] S. Karita, S. Watanabe, T. Iwata, M. Delcroix, A. Ogawa, and T. Nakatani, "Semi-supervised end-to-end speech recognition using text-to-speech and autoencoders," in *Proc. of ICASSP*, 2019.

[13] A. Rosenberg, Y. Zhang, B. Ramabhadran, Y. Jia, P. Moreno, Y. Wu, and Z. Wu, "Speech recognition with augmented synthesized speech," in *Proc. of ASRU*, 2019.

[14] L. Galescu, E. Ringger, and J. Allen, "Rapid language model development for new task domains," in *Proc. of LREC*, 1998.

[15] Y.-Y. Wang, M. Mahajan, and X. Huang, "A unified context-free grammar and n-gram model for spoken language processing," in *Proc. of ICASSP*, 2000.

[16] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

[17] J. Schalkwyk, L. Hetherington, and E. Story, "Speech recognition with dynamic grammars using finite-state transducers," in *Proc. of Eurospeech*, 2003.

[18] P. R. Dixon, C. Hori, and H. Kashioka, "A specialized WFST approach for class models and dynamic vocabulary," in *Proc. of Interspeech*, 2012.

[19] P. Aleksic, C. Allauzen, D. Elson, A. Kracun, D. M. Casado, and P. J. Moreno, "Improved recognition of contact names in voice commands," in *Proc. of ICASSP*, 2015.

[20] B. Haynor and P. S. Aleksic, "Incorporating written domain numeric grammars into end-to-end contextual speech recognition systems for improved recognition of numeric sequences," in *Proc. of ICASSP*, 2020.

[21] R. Huang, O. Abdel-hamid, X. Li, and G. Evermann, "Class LM and word mapping for contextual biasing in End-to-End ASR," in *Proc. Interspeech*, 2020.

[22] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *The Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[23] T. Mikolov, S. Kombrink, L. Burget, J. Černocký, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Proc. of ICASSP*, 2011.

[24] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," in *Proc. of ICLR*, 2017.

[25] K. Irie, S. Kumar, M. Nirschl, and H. Liao, "RADMM: Recurrent adaptive mixture model with applications to domain robust language modeling," in *Proc. of ICASSP*, 2018.

[26] E. Variani, D. Rybach, C. Allauzen, and M. Riley, "Hybrid autoregressive transducer (hat)," in *Proc. of ICASSP*, 2020.

[27] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *arXiv preprint arXiv:1808.06226*, 2018.

[28] P. F. Brown, V. J. Della Pietra, P. V. Desouza, J. C. Lai, and R. L. Mercer, "Class-based n-gram models of natural language," *Computational linguistics*, vol. 18, no. 4, pp. 467–480, 1992.

[29] A Bruguier, D Gnanapragasam, L Johnson, K Rao, and F Beaufays, "Pronunciation learning with rnn-transducers," in *Proc. of Interspeech*, 2017.

[30] A. Graves, "Sequence transduction with recurrent neural networks," in *Proc. of ICML Representation Learning Workshop*, 2012.

[31] Y. Shi, Y. Wang, C. Wu, C.-F. Yeh, J. Chan, F. Zhang, D. Le, and M. L. Seltzer, "Emformer: Efficient memory transformer based acoustic model for low latency streaming speech recognition," in *Proc. of ICASSP*, 2021.

[32] T. Kudo, "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates," in *Proc. ACL*, 2018.

[33] J. Mahadeokar, Y. Shangguan, D. Le, G. Keren, H. Su, T. Le, C. Yeh, C. Fuegen, and M. L. Seltzer, "Alignment Restricted Streaming Recurrent Neural Network Transducer," in *Proc. SLT*, 2021.