INCORPORATING WRITTEN DOMAIN NUMERIC GRAMMARS INTO END-TO-END CONTEXTUAL SPEECH RECOGNITION SYSTEMS FOR IMPROVED RECOGNITION OF NUMERIC SEQUENCES

Ben Haynor *Petar S. Aleksic*

Google, LLC

{bhaynor, apetar}@google.com

ABSTRACT

Accurate recognition of numeric sequences is crucial for many contextual speech recognition applications. For example, a user might create a calendar event and be prompted by a virtual assistant for the time, date, and duration of the event. We propose a modular and scalable solution for improved recognition of numeric sequences. We use finite state transducers built from written domain numeric grammars to increase the likelihood of hypotheses containing matching numeric entities during beam search in an end-to-end speech recognition system. Using our technique results in relative reduction in word error rate of up to 59% on a variety of numeric sequence recognition tasks (times, percentages, digit sequences, ...).

Index Terms— Speech recognition, RNN-T, end-to-end, contextual ASR, FSTs.

1. INTRODUCTION

Accurate recognition of numeric sequences such as address numbers, times, dates, and digit sequences, is needed in various automatic speech recognition (ASR) tasks [1, 2]. In certain scenarios, such as dialog, contextual information can be used to significantly improve ASR quality. For instance, a virtual assistant might ask a user what time they would like to be woken up, in which case a contextual ASR (biasing) system can boost the probability of hypotheses containing a time, as the user is very likely to respond with an utterance containing time. As end-to-end (E2E) ASR systems become competitive with conventional models [3, 4] it is crucial that they accurately recognize numeric entities in highly contextual scenarios.

Given the wide variety of numeric sequence types, some will not be well represented in E2E system training data. Therefore, it's important to have a system that can easily adapt to new numeric entity types without any need for retraining of the E2E models.

We introduce a system for biasing towards numeric entities by building finite state transducers [5] (FSTs) representing numeric grammars and boosting the probability of matching hypotheses during beam search using on-the-fly rescoring [6].

Using finite state transducers to boost contextually relevant n-grams or interpolate contextual language models has been used to introduce context into conventional ASR systems [6–9]. Conventional systems have represented linguistically similar entities [10], numeric entities [2] or semantic entities [11] as classes. Contextual ASR systems have used these representations to boost the probability of relevant classes based on contextual information [7, 12–15] for improved ASR quality on numeric sequences recognition tasks.

Recently, similar approaches have been developed to incorporate contextual information into E2E ASR systems [16–20]. Our work is based on the contextual E2E ASR system introduced in [16] and refined in [20]. This contextual E2E ASR system uses on-the-fly rescoring of contextually relevant phrases, specified as n-grams and encoded as FSTs, to incorporate contextual information into ASR. Recent techniques for improving numeric sequence recognition in E2E systems include adding synthetic numeric training data, training in the spoken domain, and post processing written results to correct errors [21].

In this paper, we describe an approach for extending a contextual E2E ASR system by adding support for numeric classes that can be used for improved ASR quality of numeric sequences in specific contextual scenarios.

We organize the paper as follows. In Section 2 we present the overall system design. In Section 3 we present our system for converting written domain numeric grammars into numeric FSTs. In Section 4 we describe the test sets used in our experiments and present all of our experimental results. Finally, in Section 5 we draw conclusions and summarize our work.

2. SYSTEM DESIGN

Our system is based on the n-gram based contextual E2E ASR system introduced in [16] and refined in [20], which we extend by adding support for numeric entities, represented as FSTs built from grammars.

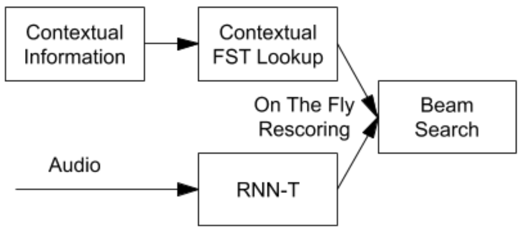

Fig. 1. Contextual E2E ASR (biasing) system. At recognition time, contextual information from the speech recognition request such as prompted entity type is used to fetch relevant contextual numeric FSTs for biasing during beam search.

We accomplish this by compiling a numeric grammar into a contextual biasing FST, C as described in Section 3. This FST is used to modify costs provided by a recurrent neural network transducer (RNN-T) [22] during beam search using on-the-fly rescoring [6]. During beam search, when computing the overall score, $S(w|H)$, of a wordpiece w, given a history H, if $w|H$ matches an arc in C we add a contextual boost $S_C(w|H)$ to the RNN-T score $S_R(w|H)$, as described in Equation 1.

$$S(w|H) = \begin{cases} S_R(w|H) - S_C(w|H)) & \text{if } (w|H) \text{ in } C \\ S_R(w|H) & \text{otherwise} \end{cases} \quad (1)$$

These FSTs are stored in an inventory. At recognition time, relevant FSTs are loaded from this inventory and used during beam search as shown in Figure 1. Which numeric FSTs to load depends on a particular contextual scenario. The FSTs can be selected by clients through an API, or by a dialog system based on a current dialog state. Biasing towards numeric entities FSTs has been shown to be useful in conventional contextual ASR systems [7], in dialog use cases, or other scenarios where certain numeric entity types are highly likely to be spoken by a user.

3. CONVERTING NUMERIC GRAMMARS INTO NUMERIC FSTS

Our system converts a written domain numeric grammar specified in OpenGrm [23] to a contextual biasing FST. For illustration purposes, we use a simple grammar accepting the strings 00 or 01, and a wordpiece model containing the strings $\{\_\_, \_\_0, \_\_1, 0, 1, 00\}$. The character __ is used to denote the start of a word. This written domain grammar, G, is shown in Figure 2.

The grapheme based grammar is converted to a wordpiece based grammar using a technique similar to the "speller" described in [16]. The speller, S, transduces from wordpieces to the constituent graphemes of that wordpiece. Using S, we create an unweighted wordpiece based acceptor grammar, U, by composing the speller S with the numeric grammar G,

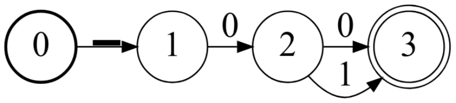

Fig. 2. An example numeric grammar FST, G, accepting 00 or 01. In this grapheme based acceptor grammar, the character __ is used to denote the start of a word.

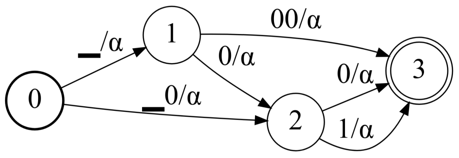

Fig. 3. Weighted wordpiece based acceptor grammar, W. It accepts all valid wordpiece based tokenizations of the numeric grammar G.

projecting onto the output, determinizing, and minimizing, as described in Equation 2.

$$U = min(det(project(S \circ G))) \quad (2)$$

Applying a constant weight, α, to each arc, results in the weighted wordpiece FST, W, shown in Figure 3. This FST accepts all valid wordpiece level tokenizations of strings in the grammar G. In this example, the number __00 can be tokenized into the wordpieces __ 0 0, __0 0, or __ 00, all of which occur in W.

The FST W increases the probability of partial matches, such as __0. Boosting partial matches was shown in [20] to reduce search errors that can occur when only boosting complete hypotheses, but makes it necessary to add failure transitions to retract boosts applied to prefixes that fail to match a complete phrase. Failure transitions are added where the weight of a failure transition is equal to the cumulative cost applied to the partial match. Therefore, all paths leading to the same state should have the same cumulative cost. The FST W contains states with ambiguous failure costs. For example, for the state 2, the path $0 \rightarrow 1 \rightarrow 2$ has a cost of 2α, while the path $0 \rightarrow 2$ has a cost of α.

To ensure unambiguous failure weights, we use a FactorWeightFst [5] to ensure that at any state, the cumulative path cost to that state is the same. The FactorWeightFst works by relabeling states by their original state ID, and the cumulative cost on a path to that state, duplicating states when cumulative costs are ambiguous. The result of applying the factor operation is the factored FST, F, shown in Figure 4. An alternative solution to the factor operation would be to create a tree FST. However, such a tree would grow exponentially large, even for simple grammars such as N-digit numbers.

Finally, failure transitions are added to F resulting in the contextual numeric FST, C, shown in Figure 5. Note that for this FST, different tokenizations of the same numeric

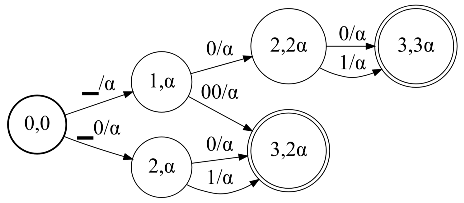

Fig. 4. The factored FST, denoted F. All paths leading to a given state have the same cumulative cost. States are labeled according to the original state ID in the unfactored FST, W, paired with the cumulative cost to reach a state.

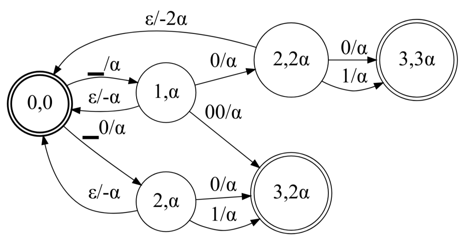

Fig. 5. The final contextual numeric FST, C, used in contextual ASR. Matching arcs have a constant cost α. Failure transitions have a cost equal to the sum of costs along any path leading to that state.

sequence get different costs. Future work might examine different weighting strategies, such as a constant cost per grapheme, rather than per arc, which would result in all paths that spell the same numeric sequence having equal cost.

One limitation of our proposal is that it only works with acyclic grammars and cannot represent arbitrary length alphanumeric sequences. In practice, this is rarely a limitation, as sequences can be restricted to some large but finite length.

The process for building and using contextually numeric FSTs can be summarized as follows:

1. Build an acyclic character based FST, G, as shown in Figure 2.
2. Construct a "speller" FST, S, mapping wordpieces to their constituent graphemes.
3. Use S and G to compute an unweighted wordpiece based acceptor grammar, U, as described in Equation 2.
4. Apply weights to U to obtain the weighted FST, W, shown in Figure 3.
5. Apply the factor operation to produce a factored FST, F, to ensure that any path leading to a given state has the same cumulative cost, as shown in Figure 4.
6. Add failure transitions with weight equal to the cumulative cost of a partial match, to obtain the contextual numeric FST, C, shown in Figure 5.
7. Use the contextual numeric FST to modify scores of matching labels during beam search as described in Equation 1.

4. EXPERIMENTAL RESULTS

In this section we describe each of the test sets used, experimental setup, metrics, and analyze the effect of contextually biasing towards numeric FSTs on ASR quality for numeric sequences.

We report word error rate (WER) and sentence error rate (SER) for each test set. The SER is the fraction of sentences containing one or more errors. We believe SER may be a more useful metric than WER to measure relative improvements on numeric entity recognition tasks, since any error is likely to make a numeric sequence unusable by downstream natural language understanding applications. We report SER rather than sentence accuracy (SACC) because relative percentage changes in SER are easier to interpret than relative changes in SACC.

To determine the effect of biasing on unrelated speech recognition tasks, each contextual FST is also used with an anti-context test set described below. This test set is decoded with and without a particular numeric FST provided as context in order to determine whether the numeric FSTs over trigger and cause quality regression on unrelated queries.

4.1. Corpora

The experiments analyzed below use various test sets in American English consisting of anonymized, manually-transcribed utterances.

In the baseline setup, experiments are run with no context provided. To evaluate the positive effect of relevant context, a contextually relevant FST is attached to every utterance in a test set containing relevant numeric entities. To evaluate the negative effect of irrelevant context, the same numeric FST is attached to a set of utterances for which the context is irrelevant, an anti-context test set.

4.1.1. Contextual Numeric Entity Test Sets

The *Time* test set consists of 501 utterances containing various standalone times, e.g. "4:30", read without any surrounding words. Such utterances might represent user responses to a prompt, "What time should I set the alarm?"

The *Percentage* test set consists of 244 utterances containing standalone percentages, e.g. "12%", read without any surrounding words.

The *Four Digits* test set consists of 522 utterances containing four-digit numbers, read without any surrounding words.

The *Ten Digits* test set consists of 512 utterances containing 10-digit numbers, read without any surrounding words.

The *Unprompted Ten Digits* test sets consists of 4,012 utterances containing 10-digit numbers uttered within context, e.g. “enter 1234567890”. This test set demonstrates that the technique improves ASR quality for more complex queries, containing words other than the numeric entities.

4.1.2. Anti-Context Test Set

The anti-context test set consists of 5,000 utterances that are unrelated to each of the prompted numeric entities that we analyze in our experiments. In anti-context experiments, each of the numeric FSTs is used individually as context in order to evaluate its effect on ASR quality of unrelated queries.

4.2. Contextual ASR Experimental Results

We measured the effect of contextual ASR using numeric FSTs on all test sets described above. The results for all test sets are shown in Table 1. We obtained ASR quality improvements on all test sets, with WER relative reduction ranging from 12.9% to 59.0%. SER improves as well on all test sets, with relative reduction ranging from 25.0% to 55.3%. The anti-context experiments showed a very small negative effect on anti-context test sets. Anti-context experiments showed WER relative increase ranging from 0.2% to 2.1% and SER relative increase ranging from 0% to 1.0%.

Model	WER [%]	Anti-context WER [%]	SER [%]	Anti-context SER [%]
Time Baseline	6.2	6.2	2.2	23.0
+ Contextual FST	5.4 (-12.9%)	6.4 (+2.1%)	1.4 (-33.3%)	23.2 (+1.0%)
Percentage Baseline	25.4	6.2	15.6	23.0
+ Contextual FST	11.1 (-56.5%)	6.3 (+1.2%)	7.0 (-55.3%)	23.1 (+0.3%)
Four Digits Baseline	23.2	6.2	14.6	23.0
+ Contextual FST	17.0 (-26.4%)	6.3 (+1.2%)	10.9 (-25.0%)	23.1 (+0.3%)
Ten Digits Baseline	40.0	6.2	26.8	23.0
+ Contextual FST	16.4 (-59.0%)	6.2 (+0.2%)	14.3 (-46.7%)	23.0 (+0.0%)
Unprompted Ten Digits Baseline	8.4	6.2	13.3	23.0
+ Contextual FST	4.7 (-44.0%)	6.2 (+0.2%)	8.7 (-34.8%)	23.0 (+0.0%)

Table 1. WER and SER for five numeric test sets and corresponding anti-context test sets. The percent changes are relative to the baseline metrics of the same test set obtained by using the same ASR system but without contextual biasing towards numeric FSTs.

4.3. Experimental results analysis

To gain a better understanding of the improvements made by the system, we inspected the effect of biasing on individual utterances. Some example improvements are shown in Table 2.

Contextual biasing fixes several types of errors. In some cases, the baseline system outputs a numeric sequence with the incorrect number of digits. In other cases, digits or digit sequences are recognized as phonetically similar words. We see similar patterns of improvements across all test sets.

The degradation on anti-context test sets is generally small. The balance between in domain test set quality gains and degradation on the anti-contxt test set can be controlled by using the weight α as shown in Figure 3.

Test set	Baseline	Contextual ASR
Percent	*90 president*	*90%*
	Not 8%	*98%*
Four Digits	*572*	*5782*
	901 ford	*9014*
Time	*Pho City*	*4:50*
	Anthony	*8:30*

Table 2. Example improvements from contextual ASR. In all examples contextual ASR corrected the hypothesis to match the transcript truth.

5. CONCLUSIONS

We presented a modular and scalable contextual E2E ASR system for recognizing numeric entities using on-the-fly rescoring of hypotheses matching written domain numeric grammars. We described a technique for conversion of grammars representing numeric entities in the written domain to FSTs that can be used to increase the probability of hypotheses matching these grammars during beam search in E2E ASR systems.

Our technique produced large ASR quality WER relative reduction ranging from 12.9% to 59.0% on five test sets. We believe that the described technique fills a previously unaddressed need of contextual recognition of numeric entities in E2E ASR systems, especially for the use cases such as, recognizing entities in response to dialog prompts from virtual assistants of interactive voice response systems.

6. ACKNOWLEDGMENTS

We would like to thank Cyril Allauzen, Pat Rondon, Zelin Wu, Cal Peyser, and David Rybach for many useful conversations.

7. REFERENCES

[1] H. Sak, F. Beaufays, K. Nakajima, and C. Allauzen, "Language model verbalization for automatic speech recognition," in *ICASSP*, 2013.

[2] L. Vasserman, V. Schogol, and K.B. Hall, "Sequence-based class tagging for robust transcription in asr," in *Interspeech*, 2015.

[3] C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, K. Gonina, N. Jaitly, B. Li, J. Chorowski, and M. Bacchiani, "State-of-the-art speech recognition with sequence-to-sequence models," in *ICASSP*, 2018.

[4] 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 *ICASSP*, 2019.

[5] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "Openfst: A general and efficient weighted finite-state transducer library," in *Proceedings of the 12th International Conference on Implementation and Application of Automata*, 2007.

[6] K. Hall, E. Cho, C. Allauzen, F. Beaufays, N. Coccaro, K. Nakajima, M. Riley, B. Roark, D. Rybach, and L. Zhang, "Composition-based on-the-fly rescoring for salient n-gram biasing," in *Interspeech*, 2015.

[7] P. Aleksic, M. Ghodsi, A. Michaely, C. Allauzen, K. Hall, B. Roark, D. Rybach, and P. Moreno, "Bringing contextual information to google speech recognition," in *Interspeech*, 2015.

[8] J. Scheiner, I. Williams, and P. Aleksic, "Voice search language model adaptation using contextual information," in *SLT*, 2016.

[9] B. Ballinger, C. Allauzen, A. Gruenstein, and J. Schalkwyk, "On-demand language model interpolation for mobile speech input," in *Interspeech*, 2010, pp. 1812–1815.

[10] P. Brown, P. deSouza, R. Mercer, V. Pietra, J. Della, and J. Lai, "Class-based n-gram models of natural language," *Computational Linguistics*, 1992.

[11] L. Velikovich, "Semantic model for fast tagging of word lattices," in *SLT*, 2016.

[12] L. Vasserman, B. Haynor, and P. Aleksic, "Contextual language model adaptation using dynamic classes," in *SLT*, 2016.

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

[14] M. Levit, A. Stolcke, R. Subba, S. Parthasarathy, S. Chang, S. Xie, T. Anastasakos, and B. Dumoulin, "Personalization of word-phrase-entity language models," in *Interspeech*, 2015.

[15] L. Velikovich, I. Williams, J. Scheiner, P. Aleksic, P. Moreno, and M. Riley, "Semantic lattice processing in contextual automatic speech recognition for google assistant," in *Interspeech*, 2018.

[16] I. Williams, A. Kannan, P. Aleksic, D. Rybach, and T. N. Sainath, "Contextual speech recognition in end-to-end neural network systems using beam search," in *Interspeech*, 2018.

[17] Z. Chen, M. Jain, Y. Wang, M. Seltzer, and C. Fuegen, "End-to-end contextual speech recognition using class language models and a token passing decoder," in *ICAASP*, 2018.

[18] A. Gandhe, A. Rastrow, and B. Hoffmeister, "Scalable language model adaptation for spoken dialogue systems," in *SLT*, 2018.

[19] A. Raju, B. Hedayatnia, L. Liu, A. Gandhe, C. Khatri, A. Metallinou, A. Venkatesh, and A. Rastrow, "Contextual language model adaptation for conversational agents," in *Interspeech*, 2018.

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

[21] C. Peyser, H. Zhang, T. N. Sainath, and Z. Wu, "Improving performance of end-to-end asr on numeric sequences," in *Interspeech*, 2019.

[22] A. Graves, "Sequence transduction with recurrent neural networks," in *ICASSP*, 2012.

[23] B. Roark, R. Sproat, C. Allauzen, M. Riley, J. Sorensen, and T. Tai, "The opengrm open-source finite-state grammar software libraries," in *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the System Demonstrations, July 10, 2012, Jeju Island, Korea*, 2012, pp. 61–66.