INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

Implementing contextual biasing in GPU decoder for online ASR

Iuliia Nigmatulina[1,2], *Srikanth Madikeri*[1], *Esaú Villatoro-Tello*[1], *Petr Motlicek*[1,3]
Juan Zuluaga-Gomez[1,4], *Karthik Pandia*[5], *Aravind Ganapathiraju*[5]

[1]Idiap Research Institute, Switzerland [2]University of Zurich, Switzerland
[3] Faculty of Information Technology, Brno University of Technology, Czech Republic
[4]LIDIAP, Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland
[5]Uniphore, India

iuliia.nigmatulina@idiap.ch

Abstract

GPU decoding significantly accelerates the output of ASR predictions. While GPUs are already being used for online ASR decoding, post-processing and rescoring on GPUs have not been properly investigated yet. Rescoring with available contextual information can considerably improve ASR predictions. Previous studies have proven the viability of lattice rescoring in decoding and biasing language model (LM) weights in offline and online CPU scenarios. In real-time GPU decoding, partial recognition hypotheses are produced without lattice generation, which makes the implementation of biasing more complex. The paper proposes and describes an approach to integrate contextual biasing in real-time GPU decoding while exploiting the standard Kaldi GPU decoder. Besides the biasing of partial ASR predictions, our approach also permits dynamic context switching allowing a flexible rescoring per each speech segment directly on GPU. The code is publicly released[1] and tested with open-sourced test sets.

Index Terms: real-time speech recognition, contextual adaptation, GPU decoding, finite-state transducers

1. Introduction

Contextual biasing of ASR has proven to be useful for many applications where prior information is available. Typically, contextual biasing in ASR works by adjusting weights of the model, or costs of words in recognition lattices, and it has been used to improve recognition of named entities (NEs), such as contacts, locations, film titles, etc. [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. Real-time decoding can work on central processing units (CPUs), as well as on graphics processing units (GPUs) that can considerably accelerate decoding [11]. Although the previous biasing methods perform well, most experiments were done for decoding on CPUs and thus are missing possible acceleration of real-time transcribing available while decoding with GPUs. This paper focuses on (1) contextual biasing in real-time decoding, (2) dynamic integration of contextual information in online GPU-based decoders, and (3) analysis of the relevance of parallelized contextual biasing.

For all experiments presented in the paper, we use Kaldi online decoders [11, 12]. In these decoders, lattice generation to pick the best output path is performed on the CPUs. Generation of real-time partial transcriptions works differently to avoid the generation of lattices, which would considerably increase the latency to obtain the final system output. Thus, a GPU-based online decoder outputs partial predictions directly selecting the current best sequence of tokens, and allows the process to be fully parallelized. This implementation, however, makes it impossible to integrate standard methods of contextual biasing directly into the online GPU decoder, as they involve lattice rescoring with FST composition. Although the implementation of lattice composition on GPUs has been proposed before [13, 14], the main challenges of the current work is to find a rescoring method without directly dealing with lattices and avoiding lattice generation for partial hypotheses.

Another challenge arises during the use of multiple contextual biases in the same decoder, where each utterance has its own biasing FST. Such a situation may arise, for instance, when another modality is providing the constantly changing contextual information for decoding[2]. It becomes prohibitively large to maintain multiple FSTs in-memory due to the nature of the composition even though the actual contextual information encoded amounts to few textual tokens. We address this challenge by keeping the contextual information independent of the original decoding graph by only storing a list of indices of the arcs to be boosted for each context. A discounting offset is added to these arcs only during decoding, thus indirectly composing the context graph with the decoding graph.

Contextual information is usually passed as a small weighted finite state transducer (WFST) created from a list of words and/or word sequences to be boosted. In this paper, we extend the previous work on contextual biasing [1, 2, 3, 7, 8, 15]. We investigate the suitability of existing biasing methods from the previous work for the GPU online decoder and propose a new approach to dynamically boost contextual information in the parallelized GPU decoding that does not require lattice generation. Our main contributions are (1) an analysis of possible ways to use contextual information in online parallelized decoding, and (2) the first publicly available implementation of dynamic contextual biasing in an online GPU decoder.

2. Online decoding on CPUs vs GPUs

2.1. Decoding on CPUs

Online decoding on CPUs (for example, Kaldi's online2-tcp-nnet3-decode-faster) is done in a similar way as offline decoding. Token and link structures are translated into OpenFst structures [16] that present an exact lattice [17]. An exact lattice contains paths, which correspond to the candidates for ASR predictions, and stores precise costs and state-level alignments. The lattice structure enables flexible post-processing with dif-

This work was supported by the Idiap&Uniphore collaboration project and partially by CRiTERIA (EC Horizon 2020, n.: 101021866) and ROXANNE (EC Horizon 2020, n.: 833635).

[1]https://github.com/idiap/contextual-biasing-on-gpus

[2]For example, the change of time, location, topic of conversation, etc. In the air traffic communication (ATC) domain, such information may come from radar data.

10.21437/Interspeech.2023-2449

ferent operations possible on lattices, such as acoustic scaling, computing the best, n-best, or oracle hypotheses, LM rescoring, lattice composition, etc. Although the lattice structure is convenient for operating with ASR output candidates before choosing the best ones, its implementation on GPUs would not be trivial. In the Kaldi GPU decoder, lattices are still created when an endpoint is reached to allow rich post-processing on CPUs [18].

2.2. Decoding on GPUs

There are many implementations of GPU decoders [19, 20, 11] and post-processing of their outputs with the CPU [20, 18, 14]. For this study, we choose to work with the standard Kaldi GPU WFST decoder [18, 11], as it is (1) open-source and (2) mostly built with Kaldi basic functions. The decoder[3] yields up to a 240x speedup over single-core CPU decoding [11]. This approach has fully parallelized decoding up to the outputs, yet its current implementation does not allow any flexible rescoring. We proposed the rescoring approach inside Kaldi GPU decoder which is fully integrated into the parallelized decoding process, with no need of lattices. It allows to asynchronously output intermediate results during online decoding without interrupting the computational process. The decoder pipeline first transfers the models (i.e. acoustic model, and $HCLG$ graph) to the GPU. The $HCLG$ graph is represented by a special structure (*cuda-fst*) on the GPU. The FST structure is represented as a set of *compressed sparse rows* (CSRs) and additional metadata, which can be efficiently traversed with direct indexing [11].

3. Methods for contextual biasing

In this section, we analyze existing and the most relevant rescoring methods, choose the best implementation strategy and mention possible limitations when working on GPUs. The common feature of all methods under consideration is that contextual information is presented as a small *biasing FST* built with the list of words and/or words sequences we want to boost.

3.1. Rescoring with lattice composition

Lattice rescoring. One of the most used ways of contextual biasing on-the-fly is lattice rescoring in the second-pass decoding [1, 2, 3, 4, 5, 6, 7, 8]. As our goal is to avoid lattice generation, this method is not suitable for us.

HCLG ° biasing_G. A rescoring approach without lattices is used in [7] and assumes boosting the $HCLG$ decoding graph before decoding. The $HCLG$ graph is composed with *biasing FST*, which leads to the target word weight adjustment directly in the decoding graph. If word sequences are boosted by this method, there is a limitation left: one can adjust the weights of only those sequences, which already exist in $HCLG$, yet, no new unknown sequences can be added. For example, if a 3-gram LM is used, only unigrams, bi-grams, and 3-grams already present in the LM can be boosted but not longer n-grams.

HCL ° G_boosted. The rescoring method proposed in [8] overcomes the limitation of the $HCLG \circ biasing\_G$ method. First, the $G.fst$ is separately modified in an iterative fashion in order (1) to adjust weights of those contextual entities, which are already present in the LM, and (2) to add new entities we want to boost, which are not present in the LM. Then, a modified $G\_boosted.fst$ is composed on-the-fly with HCL, allowing all necessary information from other ASR levels to be applied to all LM n-grams including newly added word sequences [21, 22].

3.2. Rescoring without composition.

In the GPU decoder, the $HCLG$ graph is represented on GPUs with the special *cuda-fst* structure (see 2.2). When the graph is loaded, each outgoing arc is processed by its own thread with the *load balancing expand*, generating a number of candidate tokens. The *adaptive beam* is then adjusted and used to determine which candidates are added back to the main queue for further processing [11]. GPU decoders can process multiple audio streams parallelly and it is important to enable boosting specific to an audio stream. Pre-modifying the HCLG graph in advance will result in boosting all the streams.

With a decoding graph, the rescoring approach that would be the most suitable for our goal is $HCLG \circ biasing\_G$ composition. Instead of composition, weights in the $HCLG$ could be adjusted iteratively, like for *G* in the $HCL \circ G\_boosted$ method, or by their indices that would allow flexible and less computationally expensive rescoring. As decoding on GPUs we cannot afford composition with HCL following the G rescoring, it is not possible to add unknown word sequences to the graph. Thus, the n-gram set that we can boost is always limited by the LM, like in the $HCLG \circ biasing\_G$ method. The contextual *boosting* information for this method can be passed (1) as a *biasing* FST, or (2) as a list of entities we want to boost, where all words are replaced with their IDs from the symbol table.

4. Rescoring in GPU decoder

In our implementation of rescoring, we focus on three tasks: (1) unigrams boosting, (2) word sequence boosting, (3) dynamic update of contextual information, i.e. biasing FST. Another important aspect is to make sure that our implementation is optimal and that the decoding slowdown is minimized.

4.1. Implementation

The $HCLG$ graph is represented as a set of *compressed sparse rows* (CSRs) and additional metadata stored in memory. Before CSRs are generated, the arc information from the loaded $HCLG$ graph is temporarily kept in separate vectors: for input labels IDs, output labels IDs, next state IDs, and weights. This information is further copied to the GPUs. In order to rescore on GPUs, along with the decoding FST, we load the biasing FST, whose arc information is also saved in vectors but only for those arcs that should be boosted. Algorithm 1 gives the pseudocode of the procedure to determine which arcs to boost given a list of words. The algorithm is an extension of Depth First Search to find a sequence of arcs that would generate the desired sequence. We also pay attention to the possibility that the first word in the sequence may start in the middle of the utterance. During decoding, if an arc index coincides with any token index saved from the current biasing FST, the arc's weight is adjusted by the discount factor. The boosting weight is the sum of the original arc's weight and the discount factor. The discount factor we use equals -2.0 which was empirically identified in the previous studies [8]. The process is illustrated in Figure 1.

4.2. Boosting unigrams and word sequences

For every contextual biasing FST, we keep track of indices of the arcs to boost, which are identified by Algorithm 1. In the decoding kernels, this adds an extra cost of searching if the current thread is processing an arc to be boosted. To enable faster

[3] https://github.com/kaldi-asr/kaldi/tree/master/src/cudadecoder

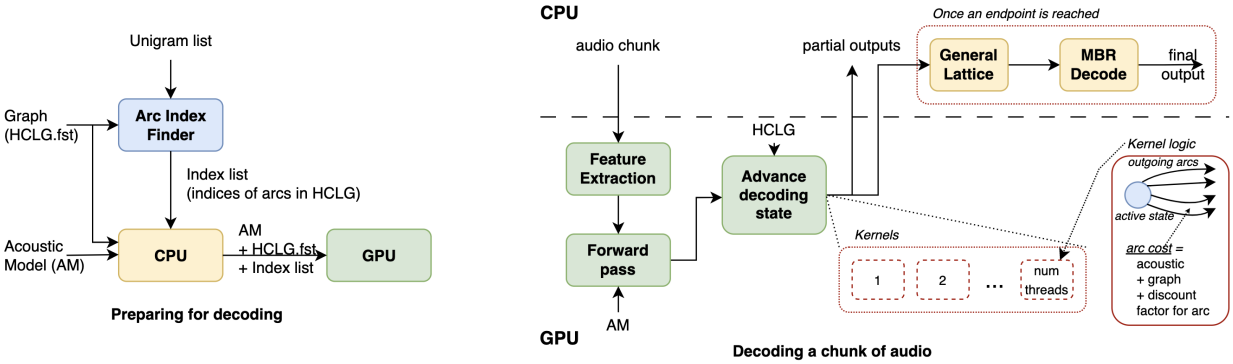

Figure 1: *Multiple stages involved in GPU decoding: initially, the models are loaded and the necessary pre-processing is done. When a chunk of audio is received, the decoding process follows with many steps split over both the CPU and the GPU.*

Algorithm 1: Pseudocode to find the arcs to be boosted given a word sequence $(w_1 w_2 ... w_k)$

Input : fst: decoding graph; $w_1 w_2 ... w_k$: word sequence to boost, N: N value of N-gram language model
Output: arcs_indices: arcs that need to be boosted
arcs_indices = Set()
statesReached = GetStatesThatOutputToken(fst, w_1)
for $t \leftarrow 2$ **to** k **do**
  prevStatesReached = statesReached
  statesReached = set()
  for $s_p \leftarrow$ *prevStatesReached* **do**
    // *DepthFirstSearchSpecial takes first edge with* w_t *and then considers only edges that output* ϵ *until* w_t *is output*
    reachableStates = DepthFirstSearchSpecial(fst, s_p, w_t)
    // *now we know we can emit the next token*
    statesReached.add(reachableStates)
    arcs_indices.Add(ArcsIndicesWithOutput(s_p, w_{t-1}))
  end
end
for $s \leftarrow$ *statesReached* **do**
  arcs_indices.Add(ArcsIndicesWithOutput(s, w_t))
end
return *arcs_indices*

search, we store the indices sorted, and perform a simple binary search. An additional complexity of $O(\log k)$ is added to each processing thread. This is negligible since k is significantly less than the total number of arcs in the graph. Compared to excessive memory requirements if storing separate decoding graphs for each context, we only store few 100s of integers.

The size of biasing FST depends on the number of entities to boost. As with the increasing number of contextual entities, the biasing effect typically goes down[4], we assume that the size of biasing FST stays small not to exceed available memory. In our experiments, the largest FST has 1013 entities (Table 2), and the number of *boosting* arcs we keep in memory per FST is significantly less than the total number of arcs since we aim to boost only the arcs related to the entities we are interested in.

4.3. Dynamic context update

To provide flexible biasing when the context is modified, we introduce the functionality of a dynamic switch between different biasing FSTs. We assume that certain context situations are anticipated in advance and the corresponding biasing FSTs are available before decoding starts. All expected biasing FSTs are pre-loaded and saved in separate vectors similar to how it is described in 4.1. Depending on the context a needed *biasing FST* indices are used to adjust the corresponding arc weights.

[4]The optimal size of biasing FST highly depends on the data; in [23], the performance began to degrade when a number of contextual entities exceeded 1000.

5. Data and experimental setup

5.1. Data

For biasing experiments we need test sets which along with text transcriptions would also have biasing list(s) with entities to boost. There are only few publicly available test sets that satisfy this criterion. One of the test sets we use is publicly available *Earning21* [24] which has been recently updated with two biasing lists based on the NER[5] [25]. The Earnings21 biasing lists contain both unigrams and word sequences, and we keep them together (in Table 1, it is categorized as sequence boosting). For decoding, we split the audios into 3-minute long segments, similar to [25]. In order to test the proposed algorithm for the case when the biasing context is always changing, we additionally use two test sets from the ATC domain. ATC conversations are usually saturated with *callsigns*[6] used to address air crafts, and *contextual data* is constantly coming from the radar that registers those callsigns of corresponding air crafts that are currently in the airspace [7, 8, 15]. One ATC test set is ATCO2 [26, 27] and another one is a publicly available 1-hour long subset of ATCO2, referred to as ATCO2-1h [28].[7] Each utterance in the ATC sets is provided with a list of callsigns to bias and with the ground truth callsign, or NO_CALLSIGN if an utterance does not contain any. ATCO2 biasing lists contain word sequences, and for the unigram boosting we converted them into lists of unique single words. All biasing lists include about 10% of OOV words. All used sets are English data; an overview including the number of biasing entities is given in Table 2.

5.2. ASR model

For the acoustic models, we use the Kaldi toolkit [12]. For the experiments on ATC data, we trained a CNN-TDNNF model with ≈1200 hours of ATC labeled data after 3-fold speed perturbation. The system follows the standard Kaldi recipe with

[5]The two biasing lists are the *oracle* and the *distractor* lists: `https://github.com/revdotcom/speech-datasets/tree/main/earnings21/bias_lists`. For our experiments, we use only the *oracle* list.

[6]Callsigns are unique identifiers for air crafts, of which the first part is an abbreviation of the airline name and the last part is a flight number that contains a digit combination and may also incorporate an additional character combination, e.g., *ryanair one sierra golf*.

[7]Website: `https://www.atco2.org/data`

Table 1: *Contextual biasing with online CPU and GPU decoders (GT is a ground truth sequence (available only for ATCO2 sets); 'partial hypotheses' are real-time model predictions; EntWER is a WER calculated for biased entities only).*

	ATCO2-4h		ATCO2-public		Earnings21		
	WER	**EntWER**	**WER**	**EntWER**	**WER**	**EntWER**	**RTFX**
Online decoding on CPU							
No biasing	32.6	36.4	24.3	26.4	21.6	59.0	7.001
Biased unigrams (partial hypotheses)	34.6	35.4	25.0	25.7	-	-	-
Biased sequences (partial hypotheses)	32.5	34.3	24.0	24.2	21.7	51.8	3.577
Biased GT (partial hypotheses)	31.0	30.4	23.1	20.3	-	-	-
Online decoding on GPU							
No biasing	32.2	36.3	24.5	26.4	21.4	60.5	26.062
Biased unigrams (at endpoints)	34.1	35.7	25.0	25.4	-	-	-
Biased sequences (at endpoints)	31.2	34.4	24.0	24.1	21.4	52.4	26.061
Biased GT (at endpoints)	30.5	30.1	23.4	21.2	-	-	-
Biased unigrams (partial hypotheses)	33.2	35.5	24.7	25.1	-	-	-
Biased sequences (partial hypotheses)	32.9	34.6	24.9	25.3	22.2	52.7	26.065
Biased GT (partial hypotheses)	30.7	29.4	23.8	21.9	-	-	-

Table 2: *Test sets with context information (number of biasing entities for ATCO2 sets† is given on average per utterance).*

Test set	**Size**	**Hours**	**Biasing entities**
ATCO2	3535 utterances	4	214†
ATCO2-public	871 utterances	1	140†
Earnings21	44 interviews	39	1013

MFCC and i-vectors features; the standard chain training is based on LF-MMI [29, 30] including one-third frame subsampling. The LM is 3-gram model trained on the same data as the acoustic model with additional text data coming from public resources such as airlines names, airports, the ICAO alphabet, and way-points in Europe. For the experiments on the Earnings21 set, we use the pre-trained chain LSTM-TDNN Kaldi model[8] with Gigaspeech-XL speech corpus [31].

5.3. Evaluation of speed

To demonstrate the lack of difference in the decoding time with VS without biasing, we measured relative decoding time with the *inverse real-time factor (RTFX)*, which is the ratio between the length of the processed audio and the decoding time. The RTFX is measured with 1 GPU NVIDIA GeForce RTX 3090, with 2K clients, and on 81 minutes of Earnings21 data, which are split into 27 utterances, each of 3 minutes in length.

6. Results

We compare WER results achieved (1) with contextual biasing on CPUs VS GPUs with lattice rescoring at *endpoints* VS dynamic biasing for *partial hypotheses* on GPUs (to see if there is performance degradation when rescoring is done without composition), (2) on GPUs with VS without applying contextual biasing (to see how the method improves the recognition). We do not compare the performance of our implementation to previous work, as there is no such results for ATCO2 sets, and biasing results on Earnings21 [25] are achieved with an End-to-End model with a different biasing approach which would be incomparable to our experiments. The results of the partial hypotheses biasing on GPUs are taken directly from the final server outputs, i.e. before it is sent for post-processing. Table 1 reports the results with utterance WER, and entities WER (EntWER) where WER is calculated only on biased entities. Comparing the performance of biasing with CPU decoder to the one on GPUs, the achieved improvement is almost the same, when rescored with lattices. Contextual biasing on GPUs always helps improve performance on the entities: e.g. EntWER 52.4% instead of 60.5%, resulting in a relative improvement of 13.4% on Earnings21. The results in utterance WERs stay the same or slightly improve when sequences are boosted. Dynamic biasing of partial hypotheses on GPUs slightly differs from the other results, as weights are modified directly in the HCLG graph instead of decoder output candidates. Overall, the performance of dynamic biasing on GPUs shows similar improvement on entities over the baseline compared to the lattice composition approach.

The main advantages of our implementation are its speed and flexibility. Decoding on GPUs allows a considerable increase in speed compared to CPUs. Adding boosting inside the GPU decoder does not slow down the decoding process with the RTFX staying almost the same: 26.06. Pre-biasing the HCLG graph in advance would lead to similar improvements but does not allow dynamic context adaptation. The main limitation is that it is not possible to add unknown word sequences to the graph and the n-gram set we can boost is always limited by the LM. In the future, the method can be also extended to WFST decoding for End-to-End models, where instead of words End-to-End model units, i.e. characters or subwords, are used.

7. Conclusion

Motivated by the high effectiveness of contextual biasing on CPUs, we proposed an algorithm and its implementation for dynamic contextual biasing on GPUs for real-time hypotheses. Given the context words and word sequences as input the method adjusts target arc weights in the decoding graph in a distributed way and without lattice generation. This approach allows fast and flexible adaptation to a current context and is a step toward closer integration between inference and decoding. A relative improvement in EntityWER of 13.4% was achieved on the Earnings21 set when biasing the target entities.

[8] https://kaldi-asr.org/models/m14

8. References

[1] 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 *Proc. of Interspeech*, 2015, pp. 468–472.

[2] 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 *Proc. of Interspeech*, 2015, pp. 1418–1422.

[3] I. Williams and P. S. Aleksic, "Rescoring-aware beam search for reduced search errors in contextual automatic speech recognition." in *Proc. of Interspeech*, 2017, pp. 508–512.

[4] J. Serrino, L. Velikovich, P. S. Aleksic, and C. Allauzen, "Contextual recovery of out-of-lattice named entities in automatic speech recognition." in *Interspeech*, 2019, pp. 3830–3834.

[5] T. Shore, F. Faubel, H. Helmke, and D. Klakow, "Knowledge-based word lattice rescoring in a dynamic context," in *Proc. of Interspeech*, 2012, pp. 1083–1086.

[6] Y. Oualil, D. Klakow, G. Szaszák, A. Srinivasamurthy, H. Helmke, and P. Motlicek, "A context-aware speech recognition and understanding system for air traffic control domain," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 404–408.

[7] M. Kocour, K. Veselỳ, A. Blatt, J. Z. Gomez, I. Szöke, J. Cernocky, D. Klakow, and P. Motlicek, "Boosting of Contextual Information in ASR for Air-Traffic Call-Sign Recognition," in *Proc. Interspeech 2021*, 2021, pp. 3301–3305.

[8] I. Nigmatulina, R. Braun, J. Zuluaga-Gomez, and P. Motlicek, "Improving callsign recognition with air-surveillance data in air-traffic communication," Idiap Research Institute. Idiap Research Institute, 2021, pp. 1–5.

[9] J. Zuluaga-Gomez, I. Nigmatulina, A. Prasad, P. Motlicek, K. Veselỳ, M. Kocour, and I. Szöke, "Contextual Semi-Supervised Learning: An Approach to Leverage Air-Surveillance and Untranscribed ATC Data in ASR Systems," in *Interspeech*, 2021, pp. 3296–3300.

[10] P. Motlicek, F. Valente, and P. N. Garner, "English spoken term detection in multilingual recordings," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[11] H. Braun, J. Luitjens, R. Leary, T. Kaldewey, and D. Povey, "Gpu-accelerated viterbi exact lattice decoder for batched online and offline speech recognition," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7874–7878.

[12] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE workshop on automatic speech recognition and understanding*, no. CONF. IEEE Signal Processing Society, 2011.

[13] A. Argueta and D. Chiang, "Composing finite state transducers on gpus," in *Proc. of the 56th Annual Meeting of the Association for Computational Linguistics (Vol. 1)*, 2018, pp. 2697–2705.

[14] K. Li, D. Povey, and S. Khudanpur, "A parallelizable lattice rescoring strategy with neural language models," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6518–6522.

[15] I. Nigmatulina, J. Zuluaga-Gomez, A. Prasad, S. S. Sarfjoo, and P. Motlicek, "A two-step approach to leverage contextual data: speech recognition in air-traffic communications," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 6282–6286.

[16] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "Openfst: A general and efficient weighted finite-state transducer library," in *International Conference on Implementation and Application of Automata*. Springer, 2007, pp. 11–23.

[17] D. Povey, M. Hannemann, G. Boulianne, L. Burget, A. Ghoshal, M. Janda, M. Karafiát, S. Kombrink, P. Motlíček, Y. Qian *et al.*, "Generating exact lattices in the wfst framework," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 4213–4216.

[18] Z. Chen, J. Luitjens, H. Xu, Y. Wang, D. Povey, and S. Khudanpur, "A gpu-based wfst decoder with exact lattice generation," in *Proc. of Interspeech*, 2018, pp. 2212–2216.

[19] A. V. Ivanov, P. L. Lange, and D. Suendermann-Oeft, "Lvcsr system on a hybrid gpu-cpu embedded platform for real-time dialog applications," in *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2016, pp. 220–223.

[20] J. Kim and I. Lane, "Accelerating large vocabulary continuous speech recognition on heterogeneous cpu-gpu platforms," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 3291–3295.

[21] T. Hori, C. Hori, Y. Minami, and A. Nakamura, "Efficient wfst-based one-pass decoding with on-the-fly hypothesis rescoring in extremely large vocabulary continuous speech recognition," *IEEE Transactions on audio, speech, and language processing*, vol. 15, no. 4, pp. 1352–1365, 2007.

[22] J. R. Novak, N. Minematsu, and K. Hirose, "Dynamic grammars with lookahead composition for wfst-based speech recognition," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[23] Z. Chen, M. Jain, Y. Wang, M. L. Seltzer, and C. Fuegen, "End-to-end contextual speech recognition using class language models and a token passing decoder," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6186–6190.

[24] M. D. Rio, N. Delworth, R. Westerman, M. Huang, N. Bhandari, J. Palakapilly, Q. McNamara, J. Dong, P. Zelasko, and M. Jette, "Earnings-21: A practical benchmark for asr in the wild," in *Interspeech*, 2021.

[25] J. Drexler Fox and N. Delworth, "Improving contextual recognition of rare words with an alternate spelling prediction model," in *Interspeech*, 2022, pp. 3914–3918.

[26] J. Zuluaga-Gomez, K. Veselỳ, I. Szöke, P. Motlicek, M. Kocour, M. Rigault, K. Choukri, A. Prasad, S. S. Sarfjoo, I. Nigmatulina *et al.*, "ATCO2 corpus: A Large-Scale Dataset for Research on Automatic Speech Recognition and Natural Language Understanding of Air Traffic Control Communications," *arXiv preprint arXiv:2211.04054*, 2022.

[27] J. Zuluaga-Gomez, I. Nigmatulina, A. Prasad, P. Motlicek, D. Khalil, S. Madikeri, A. Tart, I. Szoke, V. Lenders, M. Rigault *et al.*, "Lessons Learned in ATCO2: 5000 hours of Air Traffic Control Communications for Robust Automatic Speech Recognition and Understanding," *arXiv preprint arXiv:2305.01155*, 2023.

[28] J. Zuluaga-Gomez, A. Prasad, I. Nigmatulina, S. S. Sarfjoo, P. Motlicek, M. Kleinert, H. Helmke, O. Ohneiser, and Q. Zhan, "How does pre-trained wav2vec 2.0 perform on domain-shifted asr? an extensive benchmark on air traffic control communications," in *2022 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2023, pp. 205–212.

[29] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for asr based on lattice-free mmi." in *Interspeech*, 2016, pp. 2751–2755.

[30] S. R. Madikeri, B. K. Khonglah, S. Tong, P. Motlicek, H. Bourlard, and D. Povey, "Lattice-free maximum mutual information training of multilingual speech recognition systems." in *Interspeech*, 2020, pp. 4746–4750.

[31] G. Chen, S. Chai, G. Wang, J. Du, W.-Q. Zhang, C. Weng, D. Su, D. Povey, J. Trmal, J. Zhang *et al.*, "Gigaspeech: An evolving, multi-domain asr corpus with 10,000 hours of transcribed audio," in *Interspeech*, 2021.