INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Boosting of contextual information in ASR for air-traffic call-sign recognition

Martin Kocour[1], *Karel Veselý*[1], *Alexander Blatt*[2], *Juan Zuluaga Gomez*[3,4], *Igor Szöke*[1], *Jan "Honza" Černocký*[1], *Dietrich Klakow*[2] *and Petr Motlíček*[3]

[1]Brno University of Technology, Faculty of Information Technology, Speech@FIT, Czechia
[2]Saarland University, Saarbrücken, Germany
[3]Idiap Research Institute, Martigny, Switzerland
[4]Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland

ikocour@fit.vutbr.cz, iveselyk@fit.vutbr.cz

Abstract

Contextual adaptation of ASR can be very beneficial for multi-accent and often noisy Air-Traffic Control (ATC) speech. Our focus is call-sign recognition, which can be used to track conversations of ATC operators with individual airplanes. We developed a two-stage boosting strategy, consisting of HCLG boosting and Lattice boosting. Both are implemented as WFST compositions and the contextual information is specific to each utterance. In HCLG boosting we give score discounts to individual words, while in Lattice boosting the score discounts are given to word sequences. The context data have origin in surveillance database of OpenSky Network. From this, we obtain lists of call-signs that are made more likely to appear in the best hypothesis of ASR. This also improves the accuracy of the NLU module that recognizes the call-signs from the best hypothesis of ASR.

As part of ATCO^2 project, we collected liveatc_test_set2. The boosting of call-signs leads to 4.7% absolute WER improvement and 27.1% absolute increase of Call-Sign recognition Accuracy (CSA). Our best result of 82.9% CSA is quite good, given that the data is noisy, and WER 28.4% is relatively high. We believe there is still room for improvement.

Index terms: Air Traffic Control, Automatic Speech Recognition, Contextual Adaptation, Call-sign Recognition, Call-sign Detection, OpenSky Network.

1. Introduction

The purpose of aviation call-signs is to identify airplanes in Air Traffic Control (ATC) procedures. Many ATC messages are currently conveyed by voice over noisy VHF channel. If we had perfect call-sign recognition, we could easily track conversations of *pilots* with *ATC operators* in the shared audio channel. The tracking would be useful for post-analysis of recordings, or possibly for real-time ATC systems of the airports.

Recently, call-sign detection was an evaluation task in *Airbus Air Traffic Control challenge* [1, 2]. We redefined the task from detection to call-sign *recognition*, as we *recognize* the ICAO call-sign codes (e.g. `TVS123AB`) from the best ASR hypothesis. Then, the call-sign code can be directly interfaced to radar or other system. From the perspective of our paper, the call-sign recognition module is a black-box, and we focus on improving ATC-ASR (i.e. ASR for ATC data) by leveraging contextual information. The context we use are call-sign lists for given location and time, and these lists are queried from OpenSky Network (OSN) database [3, 4].

Several works are addressing the use of contextual information for ATC-ASR [5, 6, 7]. Shore et al. [5] introduced a lattice-rescoring mechanism, penalizing call-signs not present in the current radar situation. Schmidt et al. [6] built a grammar-based ASR, in which the search-space is limited to all command-predictions for actual radar situation. The context adaptation is continuous and integrated with on-line ASR. Later, in [7] Oualil et al. compare grammar-based and n-gram-based language modelling in ASR, showing n-grams as better. The n-grams cover well the irregularities of real ATC speech. Again, the context adaptation is continuous, and Weighted Levenshtein Distance algorithm is used to select command prediction closest to the ASR output. These works inspired us to focus on continuous adaptation and its integration into ATC-ASR with n-gram language models.

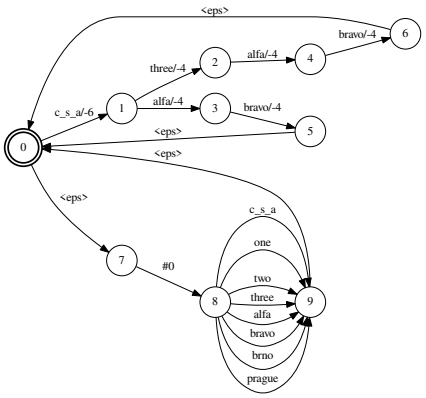

Figure 1: *Topology of WFST graph for boosting of lattices.*

Otherwise, a significant inspiration for our Lattice boosting was the work on *composition-based on-the-fly rescoring* [8], where rapid rescoring is done on unpruned pseudo-deterministic word-lattices. LM weights are adjusted for a small set of n-grams representing contextual information. Later, in *rescoring-aware beam search* [9], a secondary larger beam was introduced into the decoder generating lattices. The secondary beam is applied to the context represented as n-grams that are later biased by rescoring. The purpose is to reduce a chance that the context is pruned-out in the lattice generation. With the very same motivation, we introduce our on-the-fly HCLG graph boosting. Here, the score discounts are given to single words relevant to the context. Our technique is simpler to implement.

2. Call-sign boosting in ASR system

As *call-sign recognition* has many practical use-cases for processing ATC data, we focus on improving *Call-Sign recognition Accuracy* (CSA). We improve CSA by targeted boosting of cer-

http://dx.doi.org/10.21437/Interspeech.2021-1619

tain words, or word-strings. We give them score discounts into language model scores, which is done by means of WFST composition [8]. The boosted expressions are thus made more likely to appear in the best hypothesis of ASR. This approach is natural for Weighted Finite State Transducer (WFST) based ASR systems. And Kaldi [10] ASR systems do use OpenFst [11] for representing WFSTs.

The composition is done with a boosting graph that holds score discounts. The original language model log-scores are still used in the decoding process, as the score discounts are added as their offsets.

The boosting graph is distinct for each utterance, so we have to be aware that composition can easily become a computationally demanding operation. The complexity of WFST composition depends on numbers of states in the two operands, the number of outgoing arcs from states and a degree of non-determinism[1].

2.1. Obtaining the call-sign lists

For boosting, we need lists of candidate call-signs, which are capturing the short-term traffic situation. These can be obtained in a dynamic way from a radar system, or in a static way from a historical database of traffic monitoring. Our partner in the ATCO[2] project - OpenSky Network - provides an access to its database of surveillance data [3]. The surveillance data are collected from ADS-B receivers operated by a network of volunteers. The queries for call-sign lists are bounded both spatially and with a time-frame [12].

For evaluation sets from HAAWAII project, we use call-sign lists from radar system of the airport.

2.2. Verbalizing a call-sign

An example of the original ICAO call-sign code format from the lists is: `TVS123AB`. This can be verbalized in several ways. Our verbalization is an extension of ICAO standard [13]:

```
skytravel one two three alfa bravo
skytravel three alfa bravo
skytravel alfa bravo
skytravel one alfa bravo
skytravel one two bravo
tango victor sierra one two three alfa bravo
one two three alfa bravo
three alfa bravo
alfa bravo
```

The translation `TVS -> skytravel` is done according to a look-up table of airline designators. The rest of the code should be read as isolated numbers, and the suffix of letters is 'spelled' with ICAO alphabet. Shortening right after the airline designator is possible. Spelling of `TVS` with ICAO alphabet is also acceptable in the standard. Some common non-standard variations include shortening the airline designator `lufthansa -> hansa`, or omitting it if the situation is not ambiguous. We support also other non-standard call-sign shortenings, and number expansions of type `777 -> triple seven`. Airplanes not serving in airlines have registration number as a call-sign. The registration has a prefix that encodes country, which is spelled by ICAO alphabet (e.g. `OK` for Czech Republic, or `HB` for Switzerland).

[1] `http://www.openfst.org/twiki/bin/view/FST/ComposeDoc`

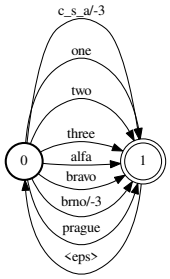

Figure 2: *Topology of WFST graph for boosting the recognition network HCLG.*

2.3. Lattice boosting

The *lattice boosting* is done as composition:

$$L' = L \circ B \,, \tag{1}$$

where L is the original lattice, and B is boosting graph. The boosting graph B is specific for each utterance.

The toy-example boosting graph in Figure 1 has a lower part with all the words in a lexicon on parallel arcs. This ensures no word sequence being dropped from the original lattice by the composition. There is also a *phi* input symbol `#0` on the 'entrance' arc to the lower part. The upper part encodes word sequences of call-signs, the score discounts -4 or -6 are on word links. We intended to experiment with a combination of per-word and per call-sign score discounts. The phi symbol allows entering lower part only if the sub-path cannot be matched by the upper part of boosting graph.

The composition is run in batch mode for whole test-set, but it could be also done on-line after finalizing the lattice. The composition is fast because both the lattices and boosting graphs are relatively small.

The word sequence must be present in the lattice in order to be boosted. The Lattice Oracle WER is computed from lattice-path that is closest to the correct transcript, and the oracle alignments can hint us of problematic words.

2.4. HCLG boosting

The *HCLG boosting* is done as composition:

$$HCLG' = HCLG \circ B \,, \tag{2}$$

where $HCLG$ [2] is the pre-compiled recognition network, and B is another type of boosting graph. The $HCLG'$ graph is used for lattice-generation, and the boosting graph B is again specific for each utterance. The composition of B with $HCLG$ graph is done on-the-fly immediately before initializing the decoder.

The toy-example boosting graph in Figure 2 boosts individual words. In the figure it is `c_s_a` and `brno` which get the score discount -3, other words have no discount. Also, note the `<eps>` back-link into state 0.

The purpose of HCLG boosting is to decrease the Lattice Oracle WER, so that the recall of call-signs in Lattice boosting increases. And, by boosting more call-signs in lattices, the final WER improves as well.

In the HCLG graph, we cannot boost word-strings as in case of using graph from Figure 1. The composition would be prohibitively slow, about 5 minutes per composition. By simplifying the boosting graph to topology from Figure 2, we already

[2] HCLG is composed from 'H' with HMM topology, 'C' for context dependency, 'L' with lexicon and 'G' for language model (grammar), more info in `https://kaldi-asr.org/doc/graph.html`

Table 1: *Audio databases for training the ASR models.*

database	hours	accents	ref.
AIRBUS	38.9	French	[14]
HIWIRE	28.7	French, Greek, Italian and Spanish	[15]
LDC ATCC	26.2	American English	[16]
MALORCA	7.9	Austrian German	[17, 18]
N4 NATO	10.7	Canadian, German, Dutch, British	[19]
ATCOSIM	10.7	German, Swiss German + French	[20]
UWB ATCC	13.2	Czech	[21, 22]
Total sum:	136.3		

Table 2: *Audio data for testing ASR and Call-sign recognition.*

test-set	hours	description
airbus_dev	1.03	custom held-out set from Airbus challenge data, mostly from 'lfbo' airport, both operator and pilot speech
malorca_vienna	1.93	test-set from project MALORCA, Vienna airport 'loww', no pilot speech
liveatc_test_set2	0.88	our own collection and manual transcription of LiveATC data, mostly Zurich airport 'lszh' plus some 'eidw' and 'katl', contains operator and pilot speech, some parts are noisy
haawaii_bikf	5.31	data from HAAWAII project, Keflavik airport 'bikf' and London Heathrow 'egll', both operator and pilot speech, 'egll' has more noise
haawaii_egll	6.85	

got an affordable increase of processing time of 20-30% on top of lattice-generation time.

We also apply epsilon-removal on B, prior to the composition, which reduces the composition run-time. In fact B could be a single state WFST right from the beginning, the second state is added for easier visualisation.

We believe that only rare, context-specific, individual words should be boosted in HCLG boosting. As we focus on call-sign recognition, we are boosting only the airline designator codewords like `skytravel`, `c_s_a` or `air_berlin`.

An alternative strategy to HCLG boosting would be to boost the `G.fst` and do the on-the-fly composition with `HCL.fst` graph as is done in Kaldi tool `nnet3-latgen-faster-lookahead`. The cascade of on-the-fly compositions $HCL \circ (G \circ B)$ would introduce some latency too. We will consider exploring this possibility as a follow-up work.

3. Experimental setup

3.1. Audio databases

For *training* the ASR, we pre-processed 7 audio databases of English ATC audio data, see Table 1. Various accents are present. Some data are from simulated scenarios (HIWIRE, N4 NATO, ATCOSIM), while other audio is from real traffic.

Table 3: *Simulating deployment of ASR to 'malorca_vienna' as a 'new' airport, WER% results.*

Training data	liveatc_test_set2	airbus_dev	malorca_vienna
with malorca	33.1	8.3	**4.7**
w/o malorca	35.1	8.4	**8.9**

Table 4: *Effect of tuning the beam-width for 'lattice boosting' and 'HCLG + lattice boosting', data-set liveatc_test_set2.*

Beams	WER%			Lattice Oracle	
	baseline	lattice boost	HCLG+lat boost	baseline	HCLG boost
b=10, lb=5	32.9	31.2	30.2	*21.4*	*19.8*
b=15, lb=8	33.1	30.0	28.8	15.2	13.9
b=20, lb=11	33.0	29.1	**28.4**	12.0	11.1
b=25, lb=13	33.1	**28.9**	**28.4**	*11.3*	*10.7*

Particularly the unification of transcripts ended up being a challenging task.

For *testing*, we use 5 different sets, see Table 2. The test-sets differ in quality of signal : 'airbus_dev', 'malorca_vienna' and 'haawaii_bikf' are clean, 'liveatc_test_set2' is quite noisy and 'haawaii_egll' contains some moderate noise. Next, 'malorca_vienna' contains no pilot speech. And further, the airports from 'liveatc_test_set2', 'haawaii_bikf' and 'haawaii_egll' are not present in training data of our ASR system.

Even though the ATC messages should follow a standard [13], we had to normalize the transcripts as follows: a) to use same ICAO alphabet, b) to use only one variant of word-splits in common expressions (e.g. 'take off' 'take-off' → 'takeoff', 'flightlevel' → 'flight level', etc.), c) to standardize the airline designators according to a "correct" table and map spaces and dashes to underscores (e.g. 'norshuttle' 'nor shuttle' → 'nor_shuttle', or 'fly niki' 'fly_niki' 'fly-niki' → 'flyniki').

3.2. Baseline ASR system

We use a 'hybrid' speech-to-text recognizer based on Kaldi [10] trained with Lattice-free MMI [23]. The neural network has 6 'conv-relu-batchnorm-layer' convolutional layers followed by 9 'tdnnf-layer' semi-orthogonal components [24]. As usual, there are two pre-final layers and two output layers: one for LF-MMI objective, the second for frame cross-entropy objective. In total, the model has 12.93 million trainable parameters, and the number of left biphone tied-states is 1680. The input features are high-resolution Mel-frequency cepstral coefficients (MFCC) with online Cepstral mean normalization (CMN). The features are extended with online i-vectors [25, 26].

Lexicon: The positive side of ATC-ASR is that the vocabulary is relatively small compared to general purpose ASR. In our case, there are 28.4k unique tokens in lexicon, out of that 15.3k are 5-letter waypoints, and 5.2k are airline designators for call-signs. We tried to create a rich vocabulary in advance to minimize the OOV problem.

We used Phonetisaurus [27] to build a grapheme to phoneme model from Librispeech lexicon [28]. We limited the vocabulary to ATC word-list gathered from 7 training databases, our test-sets, and some other pre-collected word-lists (airline designators, waypoints, airports, cities, countries, etc.).

The table of airline designators was prepared from Wikipedia page[3]. We cross-checked some items with other pub-

[3] `https://en.wikipedia.org/wiki/List_of_airline_codes`

Table 5: *Call-Sign recognition Accuracy % (CSA) and Word Error Rate % (WER) for 4 test sets and 2 types of ASR boosting: 'Lattice boosting' and 'HCLG + Lattice boosting'. The Oracle CSA is calls-sign recognition from ground truth transcripts.*

	Baseline		Lattice boost.		HCLG+Lat. boost.		Oracle
	CSA	WER	*CSA*	WER	*CSA*	WER	*CSA*
liveatc_test_set2	*53.5*	33.1	*75.6*	28.9	*80.6*	28.4	*90.0*
malorca_vienna	*84.4*	8.9	*86.5*	8.1	*88.1*	7.5	*90.5*
haawaii_bikf	-	30.6	-	29.4	-	28.9	-
haawaii_egll	-	20.8	-	19.3	-	18.8	-

lic databases. Recently, we found an FAA document[4], which could be used in future. The list of European waypoints was obtained from *traffic* [29] python project.

Language model: We use 3-gram language model built by interpolating several LMs with SRI-LM [30]. The mixing coefficients are tuned on entire 'liveatc_test_set1' (i.e. a set different from liveatc_test_set2) complemented with a fragment of 'airbus_dev' and 'malorca_vienna' test-sets. We build one LM from each training corpus transcripts (except HIWIRE and N4 NATO whose transcripts have limited variability).

An additional LM for interpolation is built from 'extra_data', i.e. a collection of: a) expanded call-signs from OSN database with 2019 world-wide traffic[5], b) all possible runway number combinations, c) European waypoints in typical idioms, and d) pre-collected word-lists previously added to lexicon.

4. Results

4.1. Deploying ASR to new airport, simulation

An ideal ATC-ASR system should generalize to a 'new' airport. In practice, the training data come from some airport, and performance for that airport is better than for some 'new' airport. We quantified this effect in Table 3.

By excluding malorca data from the training (acoustic model, language model and lexicon), the WER nearly doubles 4.7 $\rightarrow$ 8.9 for malorca_vienna test set. For other test sets, the error rate almost did not change. The malorca data consist of purely ATC operator speech, and including pilot speech would further increase the WER. Our boosting experiments are done with an ASR system that had malorca data excluded, to simulate the 'new' airport scenario.

4.2. Call-sign boosting, ASR performance, tuning beams

Next, we experiment with call-sign boosting. The call-sign words represent roughly 25% of reference transcript text. We evaluate Lattice boosting and a cascade of HCLG boosting and Lattice boosting. The liveatc_test_set2 is used to tune the beam widths and values of score discounts.

Table 4 shows a significant improvement 4.7% of WER (33.1 $\rightarrow$ 28.4) from the combination of HCLG boosting and Lattice boosting. If we do only Lattice boosting, the performance gain is little smaller (4.2%). Further widening the beams can, to some extent, compensate for not doing the HCLG boosting, but the lattices also grow larger. With 'lb=8' the liveatc_test_set2 lattices have 12MB, with 'lb=11' 89MB, and for 'lb=13' 192MB.

In first column, 'b=' stands for `--beam` and 'lb=' for `--lattice-beam`. The default values from Kaldi are 'b=15, lb=8'. Larger beams lead to better performance, but the system becomes slower. We also see the effect of HCLG boosting of airline code-words on Lattice Oracle WER. The improvements are ranging from 1.6% absolute for smaller lattices generated with narrow beams to 0.6% for wide beams.

4.3. Callsign accuracy performance

The ASR output is processed by call-sign recognition module, which is an End2End neural network that translates text directly into ICAO call-sign code like `TVS123AB`. The performance is measured as Call-Sign recognition Accuracy (CSA). The call-sign recognizer uses list of candidate call-signs as contextual information, while it still can synthesize a new call-sign not present in the list.

In Table 5, we see that WER improvements consistently translate into CSA improvements. On liveatc_test_set2 we have a huge improvement from 53.5 to 80.6. For malorca_vienna the absolute CSA improvement is smaller, nevertheless the gain from 84.4 to 88.1 removed 60.7% of the gap spanning from baseline to oracle CSA. For test-sets from HAAWAII project, we have only WER scores that show consistent improvements. For evaluation of call-sign recognition, we kept only utterances where the true call-sign was present also in the traffic monitoring data. This reduces the risk of having a wrong call-sign in the ground-truth annotation.

5. Conclusions

Inspired by other works on contextual adaptation of WFST-based ASR systems, we applied a cascade of on-the-fly HCLG boosting of individual words and Lattice boosting of word sequences. The boosted elements appear more likely as part of the best ASR hypothesis.

We focused on call-sign recognition from air-traffic control speech. Our boosting improved dramatically both the Word Error Rate and Call-sign recognition accuracy, especially for noisy test-set like liveatc_test_set2 : WER -4.7% absolute, Call-sign accuracy +27.1% absolute in Table 5. The proposed technique of giving score discounts to certain words or word sequences in ASR inference is generic and can be used in other domains.

In future, we plan to extend contextual adaptation to more types of content, for example waypoints, geographical names, or frequent expressions in local language.

6. Acknowledgements

We would like to thank to Hartmut Helmke for early feedback and DLR for providing annotated audio data from HAAWAII project. The work was supported by European Union's Horizon 2020 projects No. 864702 - ATCO2 (all auth.) and No. 884287 HAAWAII (BUT and IDIAP). Part of high-performance computation run on IT4I supercomputer and was supported by the Ministry of Education, Youth and Sports of the Czech Republic through e-INFRA CZ (ID:90140).

[4] https://www.faa.gov/documentLibrary/media/Order/7340.2J_Chg_1_dtd_10_10_19.pdf

[5] https://zenodo.org/record/3901482#.X5cK9k_0m_4

7. References

[1] T. Pellegrini, J. Farinas, E. Delpech, and F. Lancelot, "The Airbus Air Traffic Control Speech Recognition 2018 Challenge: Towards ATC Automatic Transcription and Call Sign Detection," in *Interspeech 2019, Graz, Austria, September 2019*. ISCA, 2019, pp. 2993–2997. [Online]. Available: https://doi.org/10.21437/Interspeech.2019-1962

[2] V. Gupta, L. Rebout, G. Boulianne, P. A. Ménard, and J. Alam, "CRIM's Speech Transcription and Call Sign Detection System for the ATC Airbus Challenge Task," in *Interspeech 2019, Graz, Austria, September 2019*. ISCA, 2019, pp. 3018–3022. [Online]. Available: https://doi.org/10.21437/Interspeech.2019-1131

[3] J. Sun and J. M. Hoekstra, "Integrating pyModeS and OpenSky Historical Database," in *Proceedings of the 7th OpenSky Workshop*, vol. 67, 2019, pp. 63–72.

[4] M. Schäfer, M. Strohmeier, V. Lenders, I. Martinovic, and M. Wilhelm, "Bringing up OpenSky: A large-scale ADS-B sensor network for research," in *Proceedings of the 13th IEEE/ACM International Symposium on Information Processing in Sensor Networks*. IEEE Press, 2014, pp. 83–94.

[5] T. Shore, F. Faubel, H. Helmke, and D. Klakow, "Knowledge-based word lattice rescoring in a dynamic context," in *INTERSPEECH 2012, Portland, Oregon, USA, September 2012*. ISCA, 2012, pp. 1083–1086. [Online]. Available: http://www.isca-speech.org/archive/interspeech_2012/i12_1083.html

[6] A. Schmidt, Y. Oualil, O. Ohneiser, M. Kleinert, M. Schulder, A. Khan, H. Helmke, and D. Klakow, "Context-based recognition network adaptation for improving on-line ASR in Air Traffic Control," in *2014 IEEE SLT 2014, South Lake Tahoe, NV, USA, December 2014*. IEEE, 2014, pp. 13–18. [Online]. Available: https://doi.org/10.1109/SLT.2014.7078542

[7] Y. Oualil, M. Schulder, H. Helmke, A. Schmidt, and D. Klakow, "Real-time integration of dynamic context information for improving automatic speech recognition," in *INTERSPEECH 2015, Dresden, Germany, September 2015*. ISCA, 2015, pp. 2107–2111. [Online]. Available: http://www.isca-speech.org/archive/interspeech_2015/i15_2107.html

[8] K. B. 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, Dresden, Germany, September 2015*. ISCA, 2015, pp. 1418–1422. [Online]. Available: http://www.isca-speech.org/archive/interspeech_2015/i15_1418.html

[9] I. Williams and P. S. Aleksic, "Rescoring-aware beam search for reduced search errors in contextual automatic speech recognition," in *INTERSPEECH 2017, Stockholm, Sweden, August 2017*. ISCA, 2017, pp. 508–512. [Online]. Available: http://www.isca-speech.org/archive/Interspeech_2017/abstracts/1671.html

[10] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi Speech Recognition Toolkit," in *IEEE ASRU 2011*, Dec.

[11] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: A general and efficient weighted finite-state transducer library," in *Implementation and Application of Automata, 12th International Conference, CIAA 2007, Prague, Czech Republic*.

[12] J. Zuluaga-Gomez, K. Veselý, A. Blatt *et al.*, "Automatic call sign detection: Matching air surveillance data with air traffic spoken communications," in *Proceedings of the 8th OpenSky Symposium 2020*, vol. 2020, no. 59. MDPI, 2020, pp. 1–10.

[13] "Aeronautical Telecommunications, Annex 10, Volume II," ser. Edition 6. International Civil Aviation Organization (ICAO), october 2001. [Online]. Available: https://www.icao.int/Meetings/anconf12/Document%20Archive/AN10_V2_cons%5B1%5D.pdf

[14] E. Delpech, M. Laignelet, C. Pimm, C. Raynal, M. Trzos, A. Arnold, and D. Pronto, "A real-life, French-accented corpus of air traffic control communications," in *Proceedings LREC 2018*.

[15] J. Segura, T. Ehrette, A. Potamianos, D. Fohr, I. Illina, P. Breton, V. Clot, R. Gemello, M. Matassoni, and P. Maragos, "The HIWIRE database, a noisy and non-native English speech corpus for cockpit communication," *Online. http://www. hiwire. org*, 2007.

[16] J. Godfrey, "The Air Traffic Control Corpus (ATC0) - LDC94S14A," 1994. [Online]. Available: https://catalog.ldc.upenn.edu/LDC94S14A

[17] M. Kleinert, H. Helmke, G. Siol, H. Ehr, A. Cerna, C. Kern, D. Klakow, P. Motlicek, Y. Oualil, M. Singh *et al.*, "Semi-supervised adaptation of assistant based speech recognition models for different approach areas," in *37th Digital Avionics Systems Conference (DASC)*. IEEE, 2018, pp. 1–10.

[18] A. Srinivasamurthy, P. Motlicek, I. Himawan, G. Szaszak, Y. Oualil, and H. Helmke, "Semi-supervised learning with semantic knowledge extraction for improved speech recognition in air traffic control," in *Proc. INTERSPEECH 2017*.

[19] C. Swail, L. Benarousse, E. Geoffrois, J. Grieco, R. Series, H. Steeneken, H. Stumpf, and D. Thiel, "The NATO Native and Non-Native (N4) Speech Corpus," 01 2003.

[20] K. Hofbauer, S. Petrik, and H. Hering, "The ATCOSIM corpus of non-prompted clean air traffic control speech." in *LREC*, 2008.

[21] L. Šmídl, J. Švec, D. Tihelka, J. Matoušek, J. Romportl, and P. Ircing, "Design and development of speech corpora for air traffic control training," in *Proceedings LREC 2018, Miyazaki, Japan, May 2018*. (ELRA), 2018. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2018/summaries/41.html

[22] L. Šmídl, J. Švec, D. Tihelka, J. Matoušek, J. Romportl, and P. Ircing, "Air traffic control communication (ATCC) speech corpora and their use for ASR and TTS development," *Language Resources and Evaluation*, vol. 53, no. 3, pp. 449–464, 2019.

[23] 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, San Francisco, CA, USA, September 2016*. ISCA, 2016, pp. 2751–2755. [Online]. Available: https://doi.org/10.21437/Interspeech.2016-595

[24] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohammadi, and S. Khudanpur, "Semi-orthogonal low-rank matrix factorization for deep neural networks," in *Proceedings of INTERSPEECH 2018*, 09 2018, pp. 3743–3747.

[25] V. Peddinti, G. Chen, V. Manohar, T. Ko, D. Povey, and S. Khudanpur, "JHU ASpIRE system: Robust LVCSR with TDNNS, iVector adaptation and RNN-LMS," in *2015 IEEE ASRU 2015, Scottsdale, AZ, USA, December 2015*. IEEE, 2015, pp. 539–546. [Online]. Available: https://doi.org/10.1109/ASRU.2015.7404842

[26] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors," in *2013 IEEE ASRU, Olomouc, Czech Republic, December 2013*. IEEE, 2013, pp. 55–59. [Online]. Available: https://doi.org/10.1109/ASRU.2013.6707705

[27] J. R. Novak, N. Minematsu, and K. Hirose, "Phonetisaurus: Exploring grapheme-to-phoneme conversion with joint n-gram models in the WFST framework," *Natural Language Engineering*, vol. 22, no. 6, pp. 907–938, 2016.

[28] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an ASR corpus based on public domain audio books," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[29] X. Olive and L. Basora, "A Python Toolbox for Processing Air Traffic Data: A Use Case with Trajectory Clustering," in *Proceedings of the 7th OpenSky Workshop 2019, Zurich, Switzerland, November 21-22, 2019*, ser. EPiC Series in Computing, vol. 67, 2019, pp. 73–84.

[30] A. Stolcke, "SRILM - an extensible language modeling toolkit," in *ICSLP2002 - INTERSPEECH 2002, Denver, Colorado, USA, September 16-20, 2002*, J. H. L. Hansen and B. L. Pellom, Eds. ISCA, 2002. [Online]. Available: http://www.isca-speech.org/archive/icslp_2002/i02_0901.html