





# Automatic processing pipeline for collecting and annotating air-traffic voice communication data

Martin Kocour<sup>1</sup> , Karel Veselý<sup>1</sup>, Igor Szöke<sup>1,2</sup>, Santosh Kesiraju<sup>1</sup> , Juan Zuluaga-Gomez<sup>3,4</sup> , Alexander Blatt<sup>5</sup>, Amrutha Prasad<sup>3</sup>, Iuliia Nigmatulina<sup>3</sup>, Petr Motlíček<sup>3</sup> , Dietrich Klakow<sup>4</sup>, Allan Tart<sup>6</sup>, Hicham Atassi<sup>7</sup>, Pavel Kolčárek<sup>7</sup>, Honza Černocký<sup>1</sup>, Claudia Cevenini<sup>8</sup>, Khalid Choukri<sup>9</sup>, Mickael Rigault<sup>9</sup>, Fabian Landis<sup>6</sup>, Saeed Sarfjoo<sup>3</sup>, Chloe Salamin<sup>3</sup>

<sup>1</sup> Brno University of Technology, Brno, Czechia

<sup>2</sup> ReplayWell, Brno, Czech Republic

<sup>3</sup> Idiap Research Institute, Martigny, Switzerland

<sup>4</sup> Ecole Polytechnique Fédérale de Lausanne, Switzerland

<sup>5</sup> Saarland University, Saarland Informatics Campus, Germany

<sup>6</sup> OpenSky Network

<sup>7</sup> Honeywell, Brno, Czech Republic

<sup>8</sup> Romagna Tech, Forli, Italy

<sup>9</sup> Evaluations and Language Resources Distribution Agency (ELDA), Paris, France

\* Correspondence: {ikocour,iveselyk}@fit.vut.cz

**Citation:** Kocour, M.; Veselý, K.; Blatt, A.; Szöke, I.; Kesiraju, S.; Zuluaga-Gomez, J.; Prasad, A.; Nigmatulina, I.; Motlíček, P.; et al.; Automatic processing pipeline for collecting and annotating air-traffic voice communication data. *Journal Not Specified* **2021**, *1*, 0. <https://doi.org/>

Received:  
Accepted:  
Published:

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Copyright:** © 2021 by the authors. Submitted to *Journal Not Specified* for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

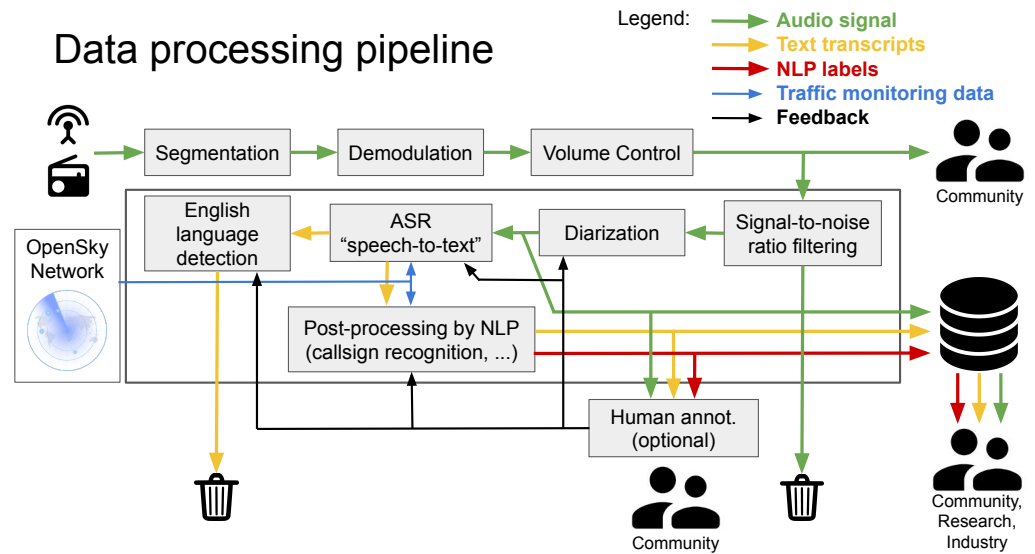
**Abstract:** This document describes our pipeline for automatic processing of the ATCO – pilot audio communication we developed as part of the ATCO<sup>2</sup> project. So far we collected two thousands of hours of audio recordings that we either pre-process for the transcribers, or we use the data for semi-supervised training. Both ways of using the collected data can further improve our pipeline by retraining our models. The proposed automatic processing pipeline is a cascade of many stand-alone components, namely: a) segmentation, b) volume control, c) signal-to-noise ratio filtering, d) diarization, e) 'speech-to-text' (ASR) module, f) English language detection, g) call-sign code recognition, h) ATCO – pilot classification and i) highlighting the commands and values. The key component of the pipeline is a speech-to-text transcription system that has to be trained with the real-world ATC data, otherwise the performance is poor. To further improve the speech-to-text performance, we apply both the semi-supervised training with our recordings, and the contextual adaptation that uses a list of plausible call-signs from surveillance data as auxiliary information. The downstream NLP/NLU tasks are important from the application point of view. These application tasks need accurate models operating on top of the real speech-to-text output, so there is a need for more data too. And creating the ATC data is the main aspiration of the ATCO<sup>2</sup> project. At the end of the project the data will be packaged and distributed by ELDA.

**Keywords:** Automatic Speech Recognition, Air Traffic Control, Contextual Adaptation, Language Identification, Named Entity Recognition, OpenSky Network

## 1. Introduction

The speech-processing tools for air-traffic data could work better if we had a large amount of reliably annotated data. However, the collection and manual annotation of air-traffic data is slow and costly. The recordings are often noisy, accented and the speech is very fast. Moreover, for the downstream tasks, we also need to label the transcripts with the air-traffic related entities. In our case those are call-signs, commands and its values (i.e. arguments of the command). Plus, other complication is that the annotators need to be experts that understand the domain.

This work presents how the whole annotation process can be efficiently accelerated by using already existing machine learning concepts. The main focus is to improve the quality of the automatic transcripts for the annotators to help them work faster.



**Figure 1.** Data-collection and data-processing pipeline.

30 The previous EU project focused on ATC speech processing was MALORCA. It  
 31 produced Active Listening Assistant (AcListant) [1], which used voice input for faster  
 32 update of a plan in the approach planning system of an airport. However, the used  
 33 speech-to-text module was tailored only to a particular airport, because the training data  
 34 was collected from two airports only. If we collect training recordings from many airports,  
 35 we believe our system will become more airport agnostic, which was a supportive  
 36 argument for collecting the data in our ATCO<sup>2</sup> project.

37 The purpose of this document is to describe the final set of tools that allows us  
 38 to pre-process and automatically transcribe the audio data that we collect in ATCO<sup>2</sup>  
 39 project. The tools are based on techniques from Signal Processing, Automatic Speech  
 40 Recognition (ASR) and Natural Language Processing (NLP), which get applied to the  
 41 air-traffic recordings. The purpose is to speed-up the process of building a large air-traffic  
 42 dataset. And, the project goals are to obtain 1000 hours of recordings with automatic  
 43 transcripts, from which 50 hours should be manually corrected by the community. The  
 44 data are planned to become accessible both for research and commercial use.

45 The overview of the data processing pipeline is in Figure 1. It consists of a) speech  
 46 pre-processing tools (segmentation, volume adjustment, discarding noisy recordings), b)  
 47 diarization (split audio per speaker), c) speech-to-text recognition, d) English language  
 48 detection, e) call-sign recognition, f) ATCO – pilot classification and g) labelling of  
 49 commands and values.

## 50 2. Data pre-processing and diarization

51 At the very beginning, the radio signal is captured by a community of feeders from  
 52 Open Sky Network by their antenna and recording device. Since the radio broadcast  
 53 is most of the time passive, i.e. the channel is silent, we apply adaptive energy-based  
 54 *segmentation* to divide the signal into segments with a voice activity. The I/Q radio  
 55 signal is converted to a waveform audio signal by a software defined radio (csdr) in the  
 56 *demodulation* block.

57 **Volume control:** the gain of the signal is increased in *volume control* in case of a weak  
 58 signal from a distant airplane. We noticed, that the speaker turns are separated by spikes,  
 59 which arise from push-to-talk control of the radio communication. Our method first  
 60 finds the positions of the spikes and then adjusts the volume of each segment separated  
 61 by spikes with a different scalar value. However, this method does not ensure that one  
 62 segment contains speech from a single speaker, as some spikes may not be detected.

63 **Signal-to-noise ratio based filtering:** our intention is to get rid of noisy segments  
 64 that are unintelligible. However, our aim is not to discard all noisy segments as moderate

noise levels in some of the training data will make the speech-to-text system more robust to noise. For estimating the Signal-to-noise ratio (SNR), we first separate the speech and non-speech parts by a Voice Activity Detection tool (VAD) described in [2]. For the use in SNR filtering, we adjusted its hyper-parameters to ensure that almost no non-speech parts are marked as speech. Since the true non-speech segments, are not recorded because of push-to-talk, we estimate SNR from the distribution of the speech signal only, which we do with the WADA-SNR (Waveform Amplitude Distribution Analysis) algorithm [3]. This method is based on the assumption that the distribution of samples from a clean speech signal is a Gamma function with a predefined shaping parameter, and the distribution of samples from additive noise is Gaussian. The method should be reliable for SNR interval 0-20dB and produces estimates of SNR values. By setting an appropriate threshold, we can discard the audio data that are too noisy, i.e. the SNR value is too low.

**Diarization:** eventually, the segments are further divided into single-speaker segments by *diarization*. The subsequent NLP tasks like call-sign recognition or command extraction need to operate on a message that comes from exactly one speaker. So, in this sense, it is strategic to identify the speaker turns already before the automatic transcription is done. Our diarization is based on Bayesian HMM clustering (VBx) [4]. The diarization is also very useful for annotators, as otherwise they would have to divide speakers manually. Currently, we use diarization just to split speech into single-speaker segments. We don't use diarization to separate ATCO – pilot speech, nor to track the utterances of the same speaker within or across recordings.

### 3. Automatic Speech Recognition

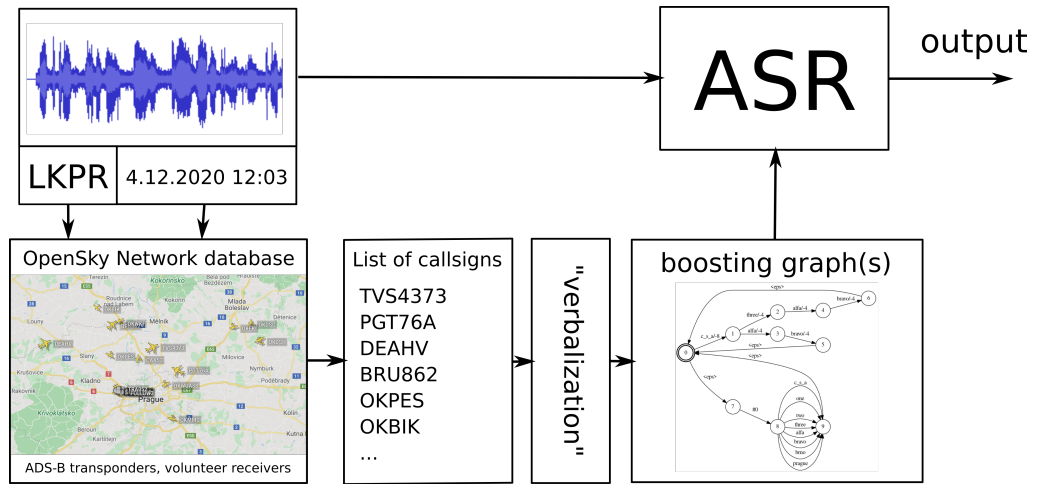
After the recordings are pre-processed, the segments are transcribed by *speech-to-text* system, that is also frequently called Automatic Speech Recognition (ASR). It is a key component in our pipeline, since it affects the performance of the downstream tasks, and we are also interested to deliver automatic transcripts of high quality. That is why we trained ASR system tailored for ATC domain. The performance of a COTS ASR would be poor on ATC data.

In this work, we used a standard *hybrid speech recognizer*, in which the temporal dynamics of speech are modeled by Hidden Markov Models (HMM). The recognizer consists of *language model* and *acoustic model*. The scores of the two models are combined together to obtain the best transcript of observed speech. The decoding itself is done with the token passing algorithm, which operates within a *recognition network* (HCLG graph). The HCLG graph is a WFST composition [5] of a graph with phone-level HMMs<sup>1</sup>  $H$ , context-dependency graph  $C$ , pronunciation lexicon graph  $L$  and language model graph  $G$ . The algorithm is using beam search heuristic to prune the improbable searched paths as the decoding progresses over time. All likely paths are then stored in a compressed format called *lattice*, i.e. a structure with timing information and pronunciation of each likely sentence. The final transcript of the segment is generated by taking 1-best hypothesis from lattice. The generated transcripts inevitably contain some errors (search errors, OOVs, etc.). Our goal is to build a system producing the least errors as possible.

For language modelling, we first created ATC text corpora from 7 ATC databases we work with (see Section 5.1 in [6]). We enriched the text corpus by list of call-signs gathered by OpenSky Network in years 2019 and 2020<sup>2</sup>. The call-signs are expanded into words by our verbalization tool, the expansion follows the ICAO standard [7] plus other common variants are generated. The expansion tool is described in our previous works [6,8,9]. We also collected a list of runways and air navigation waypoints that exist in Europe from Traffic [10], and expanded these into idiomatic contexts for language model training. It should ensure that the ASR is able to recognize all known call-signs,

<sup>1</sup> Actually, we use bi-phones, i.e. a context dependent phonemes.

<sup>2</sup> Crowdsourced air traffic data from The OpenSky Network 2020 : <https://zenodo.org/record/5644749>



**Figure 2.** Data flow diagram of boosting.

waypoints and runways. The final language model is an interpolated 3-gram in ARPA format trained with SRILM toolkit [11].

For acoustic modelling, we train a CNN-TDNN-F [12] neural network model from Kaldi with Lattice-free MMI objective function. The input features are high-resolution Mel-frequency cepstral features (MFFC) with online Cepstral mean normalization (CMN). The features are extended with online i-vectors [13]. The model produces posterior probabilities of senones (states in the HCLG recognition network) that are used by the HMM decoder.

### 3.1. Call-sign boosting

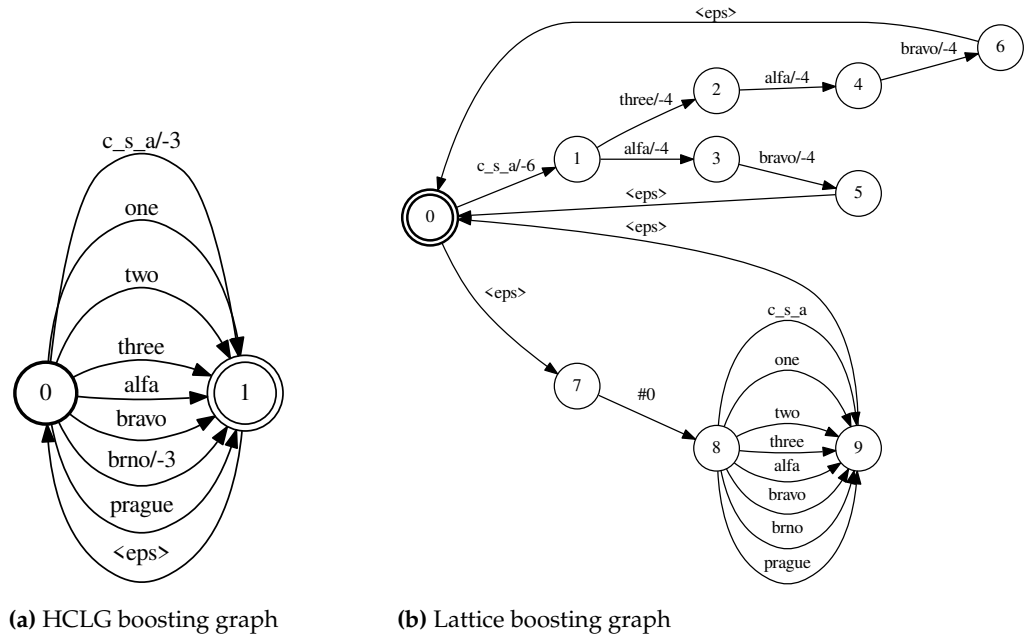
In speech-to-text, we experimented with *contextual adaptation*. This improves the ASR performance by integrating rapidly changing textual context into decoding. Since we have access to various information about the processed communication (e.g. time, location of a receiver and ADS-B surveillance data), we are able to increase a chance of correctly recognizing some specific words, in our case call-signs.

The whole process of contextual adaptation is depicted in Figure 2. For each recording, we know location and timestamp. With this info, we query the OpenSky Network database [14] for a list of call-signs using pyopensky python interface [15]. Next, the call-signs are verbalized into all its possible word sequences. Then, two boosting graphs are built from the list of verbalized call-signs, one for *HCLG boosting*, one for *lattice boosting*. These graphs are further used during the decoding, as we recently described in [8]. Note that this boosting technique allows us to do on-the-fly contextual adaptation. We have chosen to boost the call-signs, since this is the most important entity in the communication. However, the described technique, can be applied to any other entity, e.g. runway number, frequency, etc.

First, we apply the graph for HCLG boosting. This is done by WFST composition of the recognition network (HCLG) and the boosting graph  $B_{HCLG}$  :

$$HCLG' = HCLG \circ B_{HCLG} \quad (1)$$

The boosting graph  $B_{HCLG}$  contains language model score discounts for particular rare single words, that are then transferred into the recognition network  $HCLG'$ . This causes that the paths with the desired boosted words become less likely to be pruned out during decoding. So more of the boosted words are present in the lattice produced by ASR. The boosting graph  $B_{HCLG}$  is depicted in Figure 3a. Note that the topology of the HCLG boosting graph is simpler compared to lattice boosting graph in Figure 3b. This is needed for an affordable run-time of the composition with the relatively large HCLG graph.



**Figure 3.** Boosting graph examples for both a) HCLG boosting and b) Lattice boosting.

Next, after the lattice generation is finished, we apply the lattice boosting of call-signs. Similar to HCLG boosting, the lattice boosting is done by WFST composition of a boosting graph  $B_L$  with the ASR output lattice  $L$  :

$$L' = L \circ B_L \quad (2)$$

The toy example of boosting graph is shown in Figure 3b, we are boosting whole word sequences like "c\_s\_a alfa bravo". The graph 3b contains the upper part that encodes the word sequences to be boosted in a particular segment. The language model score discounts  $-4$  or  $-6$  are on the word links. The lower part contains all words from lexicon, which ensures that no words are dropped by the composition. Also, the lower part is accessed only if the partial word sequence from the lattice cannot be matched with the upper part. The HCLG boosting is done before lattice boosting just to increase a chance that the lattice contains the boosted rare words. And, the lattice boosting increases a chance of the boosted word sequence to appear in the final 1-best hypothesis, i.e. in the final transcript.

The effect of call-sign boosting is shown in Table 1. The performance was measured on our internal LiveATC test set and our public ATCO<sup>2</sup> test set for which we have the surveillance data available. With boosting we reduced the word error rate (WER) by 12.5% relative on LiveATC test set and 6.5% relative on ATCO<sup>2</sup> test set. Note that we obtained better results with both HCLG and lattice boosting than with just lattice

	LiveATC		ATCO <sup>2</sup>	
	WER [%]	CA [%]	WER [%]	CA [%]
no-boosting	35.9	46.8	21.4	77.3
HCLG-boosting	35.4	50.0	21.4	81.3
lattice-boosting	31.8	70.2	20.1	84.6
HCLG + lattice-boosting	<b>31.4</b>	<b>72.8</b>	<b>20.0</b>	<b>85.3</b>
Oracle (correct transcripts)	0.0	89.6	0.0	92.0

Table 1: Performance improvements from various call-sign boosting strategies in speech-to-text. Measured on LiveATC test set and public ATCO<sup>2</sup> test set in terms of *word error rate* (WER) and *call-sign accuracy* (CA). [ATCO<sup>2</sup> test set: <https://www.atco2.org/data>]



#		LiveATC		ATCO <sup>2</sup>	
		WER [%]	CA [%]	WER [%]	CA [%]
1	seed-system	35.9	46.8	21.4	77.3
2	SSL + gradient weighting	30.6	56.8	18.6	81.3
3	(2) + lattice boosting	27.2	73.0	17.6	85.3
4	(2) + HCLG+lattice boosting	<b>26.8</b>	<b>75.8</b>	<b>17.6</b>	<b>86.0</b>
	Oracle (correct transcripts)	0.0	89.6	0.0	92.0

Table 2: Performance improvements from semi-supervised learning (SSL) and test phase call-sign boosting. Measured on LiveATC and ATCO<sup>2</sup> test set.

161 boosting. The effect of boosting is even stronger for call-signs recognition accuracy (CA),  
 162 where it removed roughly one third of errors. The call-sign accuracy improved by 26.0 %  
 163 absolute on LiveATC test set and by 8.0 % on ATCO<sup>2</sup> test set.

### 164 3.2. Semi-supervised learning

165 We used our collected data for *semi-supervised learning* (SSL) experiments. This  
 166 improved our acoustic model by adding the untranscribed data into the training. The  
 167 untranscribed data were processed by our pipeline that filtered out noisy and non-  
 168 English data. In SSL, a seed-system is used to produce automatic transcripts and  
 169 confidences. We incorporated acoustic word confidences generated by Minimum Bayes  
 170 Risk decoding of lattices [16]. The word confidence is a probabilistic value taken from a  
 171 confusion network that has lists of candidate words for word slots.

172 We applied the word-confidences to discard 10% sentences with lowest mean  
 173 confidence. Next, we discarded all the words with confidence lower than 0.5 (5% words).  
 174 And finally, we used the word confidences to scale the gradients in the back-propagation  
 175 training. We noticed that the confidences were biased towards high values even for  
 176 incorrect words. To mitigate this, we applied power 4.0 which transformed the word-  
 177 confidences to lower values. This is rather an empirical calibration, and we can afford to  
 178 scale down some of the correct words from the automatic transcripts.

179 In our experiments we used 1190 hours of untranscribed speech, partly from  
 180 ATCO<sup>2</sup> platform, partly from LiveATC. The results are shown in Table 2, for system 2 the  
 181 WER was reduced by 14.8% for LiveATC and 13.6% for ATCO<sup>2</sup> test sets relatively. The  
 182 combination of SSL and test phase call-sign boosting achieved the best performance of  
 183 26.8% WER for LiveATC and 17.6% WER for ATCO<sup>2</sup> test sets respectively. Currently, this  
 184 is the best model we have. We further plan to start using call-sign boosting for automatic  
 185 transcripts in SSL, as we previously tried in [17].

## 186 4. English language detection

187 We have developed and deployed a suitable *English language detection* system  
 188 (ELD) [18] to discard non-English utterances in newly collected data. We tested an  
 189 acoustic based system with x-vector extractor, but then we decided to use an NLP  
 190 approach that processes ASR output with word confidences, as its performance was  
 191 better. The NLP approach can jointly use outputs from several ASR systems, which  
 192 further improves the results.

193 For the processing pipeline we integrated the NLP based English detector operating  
 194 on Czech and English ASR. The integrated English detector consists of TF-IDF for re-  
 195 weighting the accumulated soft word counts, and a Logistic regression classifier to get  
 196 the English non-English decision.

197 For each recording we extract bag-of-words statistics from the automatic transcrip-  
 198 tions generated by the ASR systems. We concatenate the word lists from lexicons of  
 199 both ASR systems. And, the statistics are accumulated from the posterior probabilities  
 200 in the bins of the confusion networks. The TF-IDF adjusts these per-word statistics by

ASR	Train data	Equal Error Rate		
		CZEN	FREN	GEEN
EN+CZ	CZEN	0.0470	0.2397	0.3433
EN+CZ	CZEN+FREN+GEEN	0.0617	0.1338	0.2602

Table 3: Performance of the English language detector with different training data (with Czech-English data CZEN, or with Czech-French-German-English data).

deweight words that appear frequently in other documents (i.e. recordings). Finally, the binary logistic regression classifier decides between the English and non-English classes.

On our evaluation set of Czech-English examples (CZEN) we achieved Equal Error Rate of 0.0470 (see Table 3). By training on more languages, the CZEN equal error rate increased to 0.0617, but for French (FREN) and German (GEEN) the error rate got smaller. If we set a threshold 0.8, we can almost completely remove the non-English utterances, while discarding only a small amount of English data.

## 5. Post-processing by NLP/NLU

The goal of the *Natural Language Understanding* (NLU) part is to extract knowledge from the text produced by the speech-to-text system. In ATCO<sup>2</sup> we focus on these tasks:

- *Call-sign recognition* (i.e. locate the call-sign, convert it to code like "DLH81J")
- *ATCO – pilot classification* (i.e. decide who is speaking in the entire utterance)
- *ATC-Entity recognition* (i.e. highlight the call-sign, command and value in text)

These tasks were selected in cooperation with our industry project partners (Honeywell, Airbus).

### 5.1. Call-sign recognition

For call-sign recognition, we use a completely neural-network based end-to-end model, which is based on BERT [19]. The model extracts the call-sign codes (e.g. DLH81J) from the speech-to-text output and the corresponding list of surveillance call-signs. We found, that this model architecture outperforms cascaded systems, where in the first step a *named entity recognizer* tags the call-sign in the transcript and in the second step a *call-sign mapper* converts the call-sign to the corresponding ICAO format.

The performance of our fully-neural *call-sign recognizer* was previously shown in Table 1 and Table 2. The performance is measured as call-sign accuracy (CA). We also see that integrating contextual information into speech-to-text engine via call-sign-boosting was essential to achieve good call-sign recognition results.

### 5.2. ATCO – pilot classification

As part of labelling the data, we aim to automatically classify the utterances based on whether it's the controller or the pilot speaking. The classifier is trained on top of the ASR-output, since there exists some disjoint vocabulary between the pilot and the controller.

The classification experiments were done using three classifiers. The first one uses a TF-IDF per-word statistics followed by a binary logistic regression, the second one is

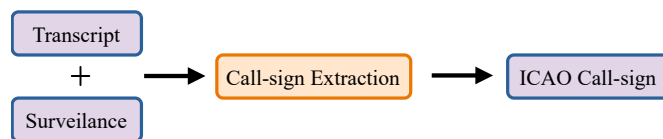


Figure 4. End-to-end model for call-sign recognition, the inputs are speech-to-text output and surveillance call-signs (list of plausible call-signs from the OSN database).

Model	Classification accuracy	
	ATCO <sup>2</sup>	LiveATC
TF-IDF + LR	77.8	76.6
CNN (no pre-training)	80.2	82.2
BERT (pre-training + fine-tuning)	91.0	87.0

Table 4: Performance of ATCO – pilot classifier.

Entity	Call-sign	Command	Value	Unknown Phraseology
LiveATC	80	52	52	34
ATCO <sup>2</sup>	89	77	68	57

Table 5: F1 Scores - on LiveATC and ATCO<sup>2</sup> test sets.

based on convolutional neural networks (CNN), and the third is based on transformer architecture called Bi-directional Encoder Representations from Transformers (BERT) [19]. The BERT model was pre-trained on masked language modelling task on large amounts of publicly available data, it was taken from Huggingface [20]. Then, we used ground truth transcripts to fine-tune the model on the sequence classification task. Around 15 thousand utterances (samples) for both ATCO and pilot classes were used for fine-tuning the model. The classification results are presented in Table 4.

The ATCO – pilot classification is not yet integrated into the data processing pipeline. The details of our work on the topic are in [21].

### 5.3. ATC-Entity recognition

Breaking down a transcript in its different components, respectively entities, can be implemented as *Named entity recognition* (NER) task. The entities of interest are call-sign <CAL>, command <COM>, value <VAL> and unknown phraseology <PHR>. The NER architecture is similar to Figure 4, but the output are NER tags in IOB format.

An example of a tagged transcript:

<COM> CLIMBING TO </COM> <VAL> FLIGHT LEVEL SEVEN ZERO </VAL>  
 <CAL> OSCAR KILO TANGO UNIFORM ROMEO </CAL>

We are currently building a database for training an ATC entity recognition network. Table 5 shows first results for a train|val|test split of 300|100|100 for the LiveATC and ATCO<sup>2</sup> test set.

## 6. Conclusion

We have successfully created an operating pipeline for processing and automatically annotating the ATCO and pilot air traffic control audio data. The pipeline discards noisy and non-English data, and generates automatic transcripts from which it extracts a call-sign code. Later, we will be able to automatically decide if it is ATCO or pilot speaking, and highlight entities in the text (call-sign, command, value). The purpose of ATCO<sup>2</sup> project is to collect a large database of ATC audio data, that will help develop better voice tools. Perhaps one day the voice tools will finally serve pilots and ATCOs.

**Author Contributions:** Conceptualization: K.V., I.S., H.Č., P.M., A.T., D.K., H.A. and P.K.; methodology: H.Č., P.M., D.K. and P.K.; software: K.V., M.K., A.B., A.P., S.S., I.N., J.Z.G. and S.K.; resources: I.S., A.T., F.L., C.S., K.Ch., M.R.; writing: M.K., K.V., A.B., S.K., C.C. and J.Z.G.; project administration: P.M.; All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by European Union’s Horizon2020 project No. 864702 - ATCO<sup>2</sup>.

**Data Availability Statement:** In this work we used 7 publicly available data, which we described in our previous work [6]. ATCO<sup>2</sup> data are freely available on <https://www.atco2.org/data>.

**Conflicts of Interest:** The authors declare no conflict of interest.



## References

1. Helmke, H.; Ohneiser, O.; Buxbaum, J.; Kern, C. Increasing ATM Efficiency with Assistant Based Speech Recognition. Twelfth USA/Europe Air Traffic Management Research and Development Seminar (ATM2017), 2017, pp. 1–10.
2. Plchot, O.; Matějka, P.; Novotný, O.; Cumani, S.; Lozano, A.D.; Slavíček, J.; Diez, M.S.; Grézl, F.; Glembek, O.; Kamsali, M.V.; Silnova, A.; Burget, L.; Ondel, L.; Kesiraju, S.; Rohdin, A.J. Analysis of BUT-PT Submission for NIST LRE 2017. Proceedings of Odyssey 2018 The Speaker and Language Recognition Workshop, 2018, pp. 47–53.
3. Kim, C.; Stern, R.M. Robust signal-to-noise ratio estimation based on waveform amplitude distribution analysis. Proc. Interspeech 2008, 2008, pp. 2598–2601. doi:10.21437/Interspeech.2008-644.
4. Landini, F.; Profant, J.; Diez, M.; Burget, L. Bayesian HMM clustering of x-vector sequences (VBx) in speaker diarization: theory, implementation and analysis on standard tasks. *Computer Speech & Language* **2022**, *71*, 101254.
5. Mohri, M.; Pereira, F.; Riley, M. Weighted finite-state transducers in speech recognition. *Computer Speech and Language* **2002**, *16*, 69–88. doi:https://doi.org/10.1006/csla.2001.0184.
6. Zuluaga-Gomez, J.; Veselý, K.; Blatt, A.; Motlíček, P.; Klakow, D.; Tart, A.; Szöke, I.; Prasad, A.; Sarfjoo, S.; Kolčárek, P.; Kocour, M.; Černocký, J.; Cevenini, C.; Choukri, K.; Rigault, M.; Landis, F. Automatic call sign detection: Matching air surveillance data with air traffic spoken communications. Proceedings of the 8th OpenSky Symposium 2020. MDPI, 2020, Vol. 2020, pp. 1–10.
7. Aeronautical Telecommunications, Annex 10, Volume II. International Civil Aviation Organization (ICAO), 2001, Edition 6.
8. Kocour, M.; Veselý, K.; Blatt, A.; Gomez, J.Z.; Szöke, I.; Černocký, J.; Klakow, D.; Motlíček, P. Boosting of contextual information in ASR for air-traffic call-sign recognition. INTERSPEECH 2021, Brno, Czechia. ISCA, 2021.
9. Zuluaga-Gomez, J.; Motlíček, P.; Zhan, Q.; Veselý, K.; Braun, R. Automatic speech recognition benchmark for air-traffic communications. Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020; Meng, H.; Xu, B.; Zheng, T.F., Eds. ISCA, 2020, pp. 2297–2301.
10. Olive, X. Traffic, a toolbox for processing and analysing air traffic data. *Journal of Open Source Software* **2019**, *4*, 1518. doi:10.21105/joss.01518.
11. Stolcke, A. SRILM - an extensible language modeling toolkit. ICSLP2002 - INTERSPEECH 2002, Denver, Colorado, USA, September 16-20, 2002; Hansen, J.H.L.; Pellom, B.L., Eds. ISCA, 2002.
12. Povey, D.; Cheng, G.; Wang, Y.; Li, K.; Xu, H.; Yarmohammadi, M.; Khudanpur, S. Semi-orthogonal low-rank matrix factorization for Deep Neural Networks. Proceedings of INTERSPEECH 2018, 2018, pp. 3743–3747. doi:10.21437/Interspeech.2018-1417.
13. Peddinti, V.; Chen, G.; Manohar, V.; Ko, T.; Povey, D.; Khudanpur, S. JHU ASPIRE system: Robust LVCSR with TDNNs, iVector adaptation and RNN-LMS. 2015 IEEE ASRU 2015, Scottsdale, AZ, USA, December 2015. IEEE, 2015, pp. 539–546.
14. Schäfer, M.; Strohmeier, M.; Lenders, V.; Martinovic, I.; Wilhelm, M. Bringing up OpenSky: A large-scale ADS-B sensor network for research. Proceedings of the 13th IEEE/ACM International Symposium on Information Processing in Sensor Networks. IEEE Press, 2014, pp. 83–94.
15. Sun, J.; Hoekstra, J.M. Integrating pyModeS and OpenSky historical database. Proceedings of the 7th OpenSky Workshop, 2019, Vol. 67, pp. 63–72.
16. Xu, H.; Povey, D.; Mangu, L.; Zhu, J. Minimum Bayes Risk decoding and system combination based on a recursion for edit distance. *Computer Speech and Language* **2011**, *25*, 802–828. doi:https://doi.org/10.1016/j.csl.2011.03.001.
17. Zuluaga-Gomez, J.; Nigmatulina, I.; Prasad, A.; Motlicek, P.; Veselý, K.; Kocour, M.; Szöke, I. Contextual semi-supervised learning: An approach to leverage air-surveillance and untranscribed ATC data in ASR systems. Proc. Interspeech 2021, 2021, pp. 3296–3300. doi:10.21437/Interspeech.2021-1373.
18. Szöke, I.; Kesiraju, S.; Novotný, O.; Kocour, M.; Veselý, K.; Černocký, J. Detecting English speech in the air traffic control voice communication. INTERSPEECH 2021, Brno, Czechia. ISCA, 2021.
19. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* **2018**.
20. Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Le Scao, T.; Gugger, S.; Drame, M.; Lhoest, Q.; Rush, A. Transformers: State-of-the-art natural language processing. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations; Association for Computational Linguistics: Online, 2020; pp. 38–45. doi:10.18653/v1/2020.emnlp-demos.6.
21. Zuluaga-Gomez, J.; Sarfjoo, S.S.; Prasad, A.; Nigmatulina, I.; Motlicek, P.; Ohneiser, O.; Helmke, H. BERTTraffic: A robust BERT-based approach for speaker change detection and role identification of air-traffic communications. *arXiv preprint arXiv:2110.05781* **2021**.