Proceeding Paper

Automatic Processing Pipeline for Collecting and Annotating Air-Traffic Voice Communication Data [†]

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

[1] Speech@FIT, Faculty of Information Technology, Brno University of Technology, 61200 Brno, Czech Republic; szoke@fit.vut.cz (I.S.); kesiraju@fit.vut.cz (S.K.); cernocky@fit.vut.cz (J.Č.)

[2] ReplayWell, 61200 Brno, Czech Republic

[3] Idiap Research Institute, 1920 Martigny, Switzerland; juan-pablo.zuluaga@idiap.ch (J.Z.-G.); amrutha.prasad@idiap.ch (A.P.); iuliia.nigmatulina@idiap.ch (I.N.); petr.motlicek@idiap.ch (P.M.); saeed.sarfjoo@idiap.ch (S.S.); chloe.salamin@idiap.ch (C.S.)

[4] Ecole Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland

[5] Department of Language Science & Technology, Saarland Informatics Campus, Saarland University, 66123 Saarbrücken, Germany; ablatt@lsv.uni-saarland.de (A.B.); dietrich.klakow@lsv.uni-saarland.de (D.K.)

[6] OpenSky Network, 3400 Burgdorf, Switzerland; tart@opensky-network.org (A.T.); landis@opensky-network.org (F.L.)

[7] Honeywell, 62700 Brno, Czech Republic; hicham.atassi@honeywell.com (H.A.); pavel.kolcarek@honeywell.com (P.K.)

[8] Romagna Tech, 47121 Forli, Italy; claudia.cevenini@studiocevenini.it

[9] Evaluations and Language Resources Distribution Agency (ELDA), 75013 Paris, France; choukri@elda.org (K.C.); mickael@elda.org (M.R.)

[*] Correspondence: ikocour@fit.vut.cz (M.K.); iveselyk@fit.vut.cz (K.V.)

[†] Presented at the 9th OpenSky Symposium, Brussels, Belgium, 18–19 November 2021.

Abstract: This document describes our pipeline for automatic processing of ATCO pilot audio communication we developed as part of the ATCO^2 project. So far, we collected two thousand hours of audio recordings that we either preprocessed for the transcribers or used for semi-supervised training. Both methods of using the collected data can further improve our pipeline by retraining our models. The proposed automatic processing pipeline is a cascade of many standalone components: (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 commands and values. The key component of the pipeline is a speech-to-text transcription system that has to be trained with real-world ATC data; otherwise, the performance is poor. In order to further improve speech-to-text performance, we apply both semi-supervised training with our recordings and the contextual adaptation that uses a list of plausible callsigns from surveillance data as auxiliary information. Downstream NLP/NLU tasks are important from an application point of view. These application tasks need accurate models operating on top of the real speech-to-text output; thus, there is a need for more data too. Creating ATC data is the main aspiration of the ATCO^2 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 processes of air-traffic data are slow and costly. The recordings are often noisy, accented, and

speech is very fast. Moreover, for downstream tasks, we also need to label the transcripts with air-traffic related entities. In our case, those are callsigns, commands and its values (i.e., arguments of the command). In addition, other complications include the need for annotators to be experts who understand the domain.

This study presents how the entire 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 in order to help them work faster.

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

The purpose of this document is to describe the final set of tools that allows us to pre-process and automatically transcribe the audio data that we collect in ATCO^2 project. The tools are based on techniques from Signal Processing, Automatic Speech Recognition (ASR) and Natural Language Processing (NLP), which are then applied to air-traffic recordings. The purpose is to speedup the process of building a large air-traffic dataset. Moreover, the project goals aim to obtain 1000 h of recordings with automatic transcripts, from which 50 h should be manually corrected by the community. The data are planned to become accessible both for research and commercial use.

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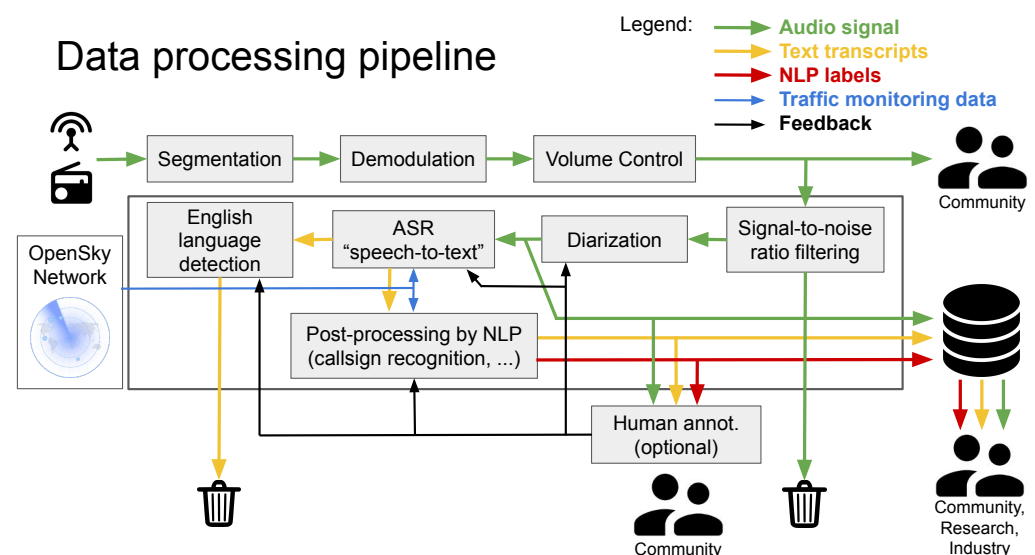

Figure 1. Data collection and data-processing pipeline.

2. Data Pre-Processing and Diarization

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

Volume control: The gain of the signal increased in *volume control* in case of a weak signal from a distant airplane. We noticed, that the speaker turns are separated by spikes,

which arise from push-to-talk control of the radio communication. Our method first finds the positions of the spikes and then adjusts the volume of each segment separated by spikes with a different scalar value. However, this method does not ensure that one segment contains speech from a single speaker, as some spikes may not be detected.

Signal-to-noise ratio based filtering: Our intention is to remove of noisy segments that are unintelligible. However, our aim is not to discard all noisy segments as moderate noise levels in some of the training data will render 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 perform 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 the SNR interval 0–20 dB and produces estimates of SNR values. By setting an appropriate threshold, we can discard 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 such as call-sign recognition or command extraction need to operate on a message that comes from exactly one speaker. Thus, in this sense, it is strategic to identify speaker turns before automatic transcription is performed. Our diarization is based on Bayesian HMM clustering (VBx) [4]. Diarization is also very useful for annotators; otherwise, they would have to divide speakers manually. Currently, we use diarization only to split speech into single-speaker segments. We do not 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, which 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 in delivering automatic transcripts of high quality. That is why we trained an ASR system tailored for ATC domain. The performance of a COTS ASR would be poor on ATC data.

In this study, 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 performed 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 (we used biphones, i.e., a context dependent phonemes) H, context-dependency graph C, pronunciation lexicon graph L and language model graph G. The algorithm uses a beam search heuristic to prune 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 the best hypothesis from the lattice. The generated transcripts inevitably contain some errors (search errors, OOVs, etc.). Our goal is to build a system producing the least amount of errors possible.

For language modelling, we first created ATC text corpora from seven ATC databases we worked with (see Section 5.1 in [6]). We enriched the text corpus by a list of callsigns gathered by OpenSky Network in years 2019 and 2020 (crowdsourced air traffic data from the OpenSky Network 2020: https://zenodo.org/record/5644749, accessed in February 2021). The call-signs are expanded into words by our verbalization tool, and the expansion

follows the ICAO standard [7], and other common variants are generated. The expansion tool is described in our previous studies [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 ASR is able to recognize all known callsigns, 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 the 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 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 the chance of correctly recognizing some specific words; in our case, these were the callsigns.

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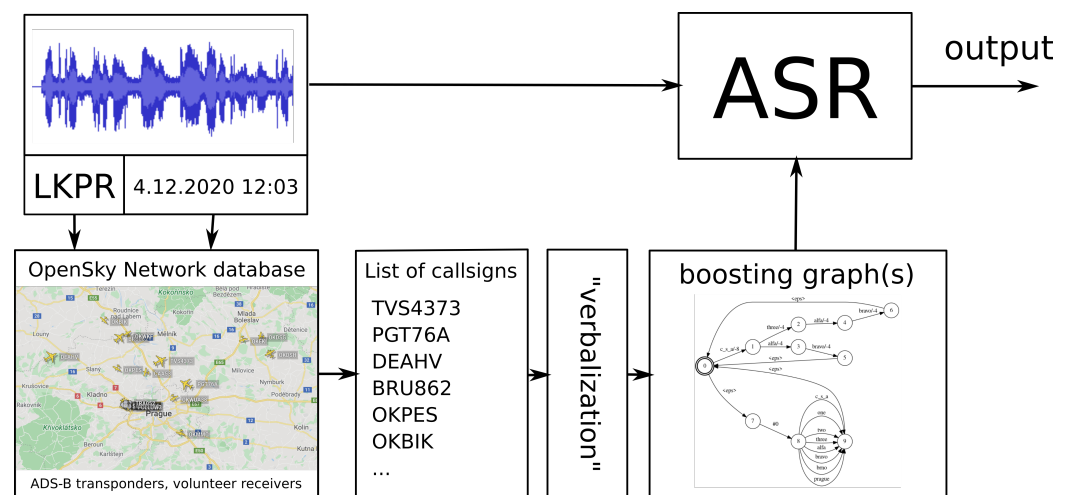

Figure 2. Data flow diagram of boosting.

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

$$HCLG' = HCLG \circ B_{HCLG} \tag{1}$$

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 the paths with desired boosted words to become less likely to be pruned out during decoding. Thus, 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 runtime of the composition with the relatively large HCLG graph.
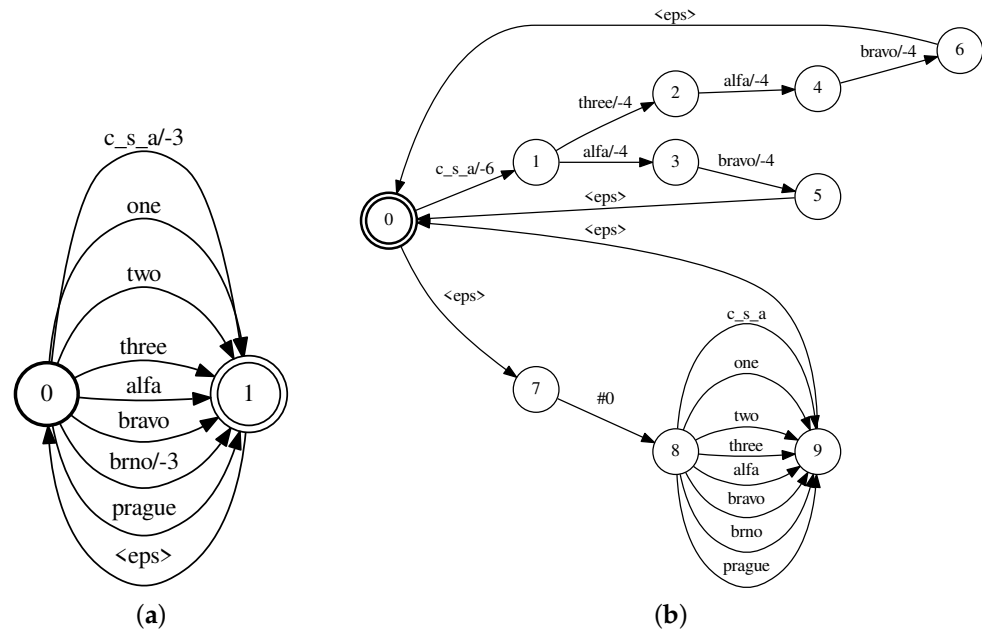

Figure 3. Boosting graph examples for both (**a**) HCLG boosting and (**b**) Lattice boosting.

Next, after lattice generation is completed, we apply lattice boosting of callsigns. Similarly to HCLG boosting, lattice boosting is performed by WFST composition of a boosting graph B_L with the ASR output lattice L.

$$L' = L \circ B_L \tag{2}$$

The toy example of boosting graph is shown in Figure 3b. We are boosting entire word sequences such as `"c_s_a alfa bravo"`. Figure 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. Moreover, the lower part is accessed only if the partial word sequence from the lattice cannot be matched with the upper part. HCLG boosting is performed before lattice boosting only to increase a chance that the lattice contains the boosted rare words. Moreover, lattice boosting increases the chance of the boosted word sequence to appear in the final 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^2 test set for which we have surveillance data available. With boosting, we reduced the word error rate (WER) by 12.5% relative to the LiveATC test set and 6.5% relative to the ATCO^2 test set. Note that we obtained better results with both HCLG and lattice boosting than with only lattice boosting. The effect of boosting is even stronger for call-sign recognition accuracy (CA), where it removed roughly one-third of errors. Call-sign accuracy improved by 26.0 % on the LiveATC test set and by 8.0 % on ATCO^2 test set.

Table 1. Performance improvements from various call-sign boosting strategies in speech-to-text. Measured on LiveATC test set and public ATCO^2 test set in terms of *word error rate* (WER) and *call-sign accuracy* (CA) (ATCO^2 test set: https://www.atco2.org/data, accessed in August 2021).

	LiveATC		ATCO^2	
	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	31.4	72.8	20.0	85.3
Oracle (correct transcripts)	0.0	89.6	0.0	92.0

3.2. Semi-Supervised Learning

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

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

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

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

#		LiveATC		ATCO^2	
		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	26.8	75.8.	17.6.	86.0
	Oracle (correct transcripts)	0.0	89.6	0.0	92.0

4. English Language Detection

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

For the processing pipeline, we integrated the NLP-based English detector operating on Czech and English ASR. The integrated English detector consists of TF-IDF for re-weighting the accumulated soft word counts, and a logistic regression classifier was used to obtain English non-English decisions.

For each recording, we extracted bag-of-words statistics from the automatic transcriptions generated by ASR systems. We concatenate word lists from lexicons of both ASR systems. Moreover, statistics are accumulated from the posterior probabilities in the bins of confusion networks. TF-IDF adjusts these per-word statistics by deweighting words that appear frequently in other documents (i.e., recordings). Finally, the binary logistic regression classifier decides between English and non-English classes.

On our evaluation set of Czech–English examples (CZEN), we achieved an 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 became smaller. If we set a threshold of 0.8, we can almost completely remove non-English utterances, while discarding only a small amount of English data.

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

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

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 ATCO², we focus on these tasks:

- *Call-sign recognition* (i.e., locate the call-sign and convert it to code such as "DLH81J");
- *ATCO—pilot classification* (i.e., decide who is speaking in the entire utterance);
- *ATC-Entity recognition* (i.e., highlight the callsign, 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 complete neural-network based end-to-end model, which is based on BERT [19]. The model extracts 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 callsign 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 Tables 1 and 2. The performance is measured as call-sign accuracy (CA). We also observe that integrating contextual information into speech-to-text engine via call-sign boosting was essential for achieving good call-sign recognition results.

5.2. ATCO—Pilot Classification

As part of labelling the data, we aim to automatically classify utterances based on whether it is 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 performed using three classifiers. The first one uses a TF-IDF per-word statistics followed by binary logistic regression, the second one is 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, and it was taken from Huggingface [20]. Then, we used ground truth transcripts to finetune the model on the sequence classification task. Around 15 thousand utterances (samples) for both ATCO and pilot classes were used for finetuning the model. The classification results are presented in Table 4.

Table 4. Performance of ATCO pilot classifier.

	Classification Accuracy	
Model	**ATCO^2**	**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

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

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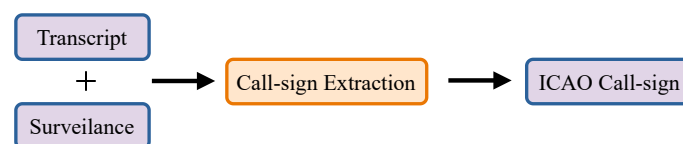

Figure 4. End-to-end model for call-sign recognition, the inputs are speech-to-text output and surveillance callsigns (list of plausible callsigns from the OSN database).

An example of a tagged transcript is as follows.

```
<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 the first results for a `train|val|test` split of 300|100|100 for the LiveATC and ATCO^2 test set.

Table 5. F1 Scores on LiveATC and ATCO^2 test sets.

Entity	Callsign	Command	Value	Unknown Phraseology
LiveATC	80	52	52	34
ATCO^2	89	77	68	57

6. Conclusions

We have successfully created an operating pipeline for processing and automatically annotating 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 callsign code. Later, we will be able to automatically decide if it is the ATCO or pilot speaking and highlight entities in the text (callsign, command and value). The purpose of the ATCO^2 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.

References

1. Helmke, H.; Ohneiser, O.; Buxbaum, J.; Kern, C. Increasing ATM Efficiency with Assistant Based Speech Recognition. In Proceedings of the Twelfth USA/Europe Air Traffic Management Research and Development Seminar (ATM2017), Seattle, WA, USA, 26–30 June 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.; et al. Analysis of BUT-PT Submission for NIST LRE 2017. In Proceedings of the Odyssey 2018 The Speaker and Language Recognition Workshop, Les Sables d'Olonne, France, 26–29 June 2018; pp. 47–53.
3. Kim, C.; Stern, R.M. Robust signal-to-noise ratio estimation based on waveform amplitude distribution analysis. In Proceedings of the Interspeech 2008, Brisbane, Australia, 22–26 September 2008; pp. 2598–2601. [CrossRef]
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. *Comput. Speech Lang.* **2022**, *71*, 101254. [CrossRef]
5. Mohri, M.; Pereira, F.; Riley, M. Weighted finite-state transducers in speech recognition. *Comput. Speech Lang.* **2002**, *16*, 69–88. [CrossRef]
6. Zuluaga-Gomez, J.; Veselý, K.; Blatt, A.; Motlíček, P.; Klakow, D.; Tart, A.; Szőke, I.; Prasad, A.; Sarfjoo, S.; Kolčárek, P.; et al. Automatic call sign detection: Matching air surveillance data with air traffic spoken communications. *Proceedings* **2020**, *59*, 14. [CrossRef]
7. *Aeronautical Telecommunications, Annex 10, Volume II*, 6th ed.; International Civil Aviation Organization (ICAO): Montreal, ON, Canada, 2001.
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. In Proceedings of the INTERSPEECH 2021 (ISCA), Brno, Czech Republic, 30 August–3 September, 2021.
9. Zuluaga-Gomez, J.; Motlíček, P.; Zhan, Q.; Veselý, K.; Braun, R. Automatic speech recognition benchmark for air-traffic communications. In Proceedings of the Interspeech 2020, 21st Annual Conference of the International Speech Communication Association (ISCA), Virtual Event, Shanghai, China, 25–29 October 2020; Meng, H., Xu, B., Zheng, T.F., Eds.; International Speech Communication Association: Shanghai, China, 2020; pp. 2297–2301.
10. Olive, X. Traffic, a toolbox for processing and analysing air traffic data. *J. Open Source Softw.* **2019**, *4*, 1518. [CrossRef]
11. Stolcke, A. SRILM—An extensible language modeling toolkit. In Proceedings of the ICSLP2002—INTERSPEECH 2002 (ISCA), Denver, CO, USA, 16–20 September 2002; Hansen, J.H.L., Pellom, B.L., Eds.; International Speech Communication Association: Denver, CO, USA, 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. In Proceedings of the INTERSPEECH 2018, Hyderabad, India, 2–6 September 2018; pp. 3743–3747. [CrossRef]
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. In Proceedings of the 2015 IEEE ASRU, Scottsdale, AZ, USA, 13–17 December 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. In Proceedings of the 13th IEEE/ACM International Symposium on Information Processing in Sensor Networks, Berlin, Germany, 15–17 April 2014; pp. 83–94.
15. Sun, J.; Hoekstra, J.M. Integrating pyModeS and OpenSky historical database. In Proceedings of the 7th OpenSky Workshop, Zurich, Switzerland, 21–22 November 2019; Voume 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. *Comput. Speech Lang.* **2011**, *25*, 802–828. [CrossRef]
17. Zuluaga-Gomez, J.; Nigmatulina, I.; Prasad, A.; Motlicek, P.; Veselỳ, K.; Kocour, M.; Szöke, I. Contextual semi-supervised learning: An approach to leverage air-surveillance and untranscribed ATC data in ASR systems. In Proceedings of the Interspeech 2021, Brno, Czech Republic, 30 August–3 September 2021; pp. 3296–3300. [CrossRef]

18. Szöke, I.; Kesiraju, S.; Novotný, O.; Kocour, M.; Veselý, K.; Černocký, J. Detecting English speech in the air traffic control voice communication. In Proceedings of the INTERSPEECH 2021 (ISCA), Brno, Czech Republic, 30 August–3 September 2021.
19. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
20. Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. Transformers: State-of-the-art natural language processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Online, 16–20 November 2020; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 38–45. [CrossRef]
21. Zuluaga-Gomez, J.; Sarfjoo, S.S.; Prasad, A.; Nigmatulina, I.; Motlicek, P.; Ohneiser, O.; Helmke, H. BERTraffic: A robust BERT-based approach for speaker change detection and role identification of air-traffic communications. *arXiv* **2021**, arXiv:2110.05781.