Machine Learning of Air Traffic Controller Command Extraction Models for Speech Recognition Applications

Hartmut Helmke, Matthias Kleinert, Oliver Ohneiser, Heiko Ehr, Shruthi Shetty
Institute of Flight Guidance, Department Controller Assistance, German Aerospace Center (DLR)
Lilienthalplatz 7, 38108 Braunschweig, Germany
Email: {hartmut.helmke, matthias.kleinert, oliver.ohneiser, heiko.ehr, shruthi.shetty}@dlr.de

***Abstract*—**Increasing digitization and automation is a widely accepted method to cope with the challenges of constantly increasing air traffic. The analogue communication of air traffic controllers (ATCo) to pilots has been excluded so far from the digitization process. However, the content of this communication is of decisive importance for various automation systems. Although Assistant Based Speech Recognition (ABSR) has recently significantly improved the recognition performance and, therefore, enables the digitization of ATCo-pilot-communication, its adaptation to other airports is a critical and costly process, This is even more important, if ATCos tend to deviate from the published ICAO phraseology: "start reducing to two fifty" instead of "reduce two five zero knots" is just an example. User acceptance requires that these deviations are also correctly recognized. Therefore, this paper presents an approach, which automatically learns a so-called *Command Extraction Model* from labelled controller utterances. The initial *Command Extraction Model* without learning only covers 60% of the commands, whereas the automatically learned *Command Extraction Model* covers more than 98%. With just six hours of training data we could achieve 94%.

Keywords—Automatic Speech Recognition, Machine Learning, Annotation, Ontology, Controller Command Extraction Model

I. INTRODUCTION

The constantly increasing air traffic creates more and more challenges concerning safety, capacity, efficiency, and environmental performance for air traffic management (ATM). These challenges in addition to the cost pressure are the key drivers for future developments in ATM. Increasing digitization and automation is a widely accepted methodical answer to cope with these challenges, as also addressed by SESAR [1] and NextGen [2] programs. The conversion of analogue data to machine-processable formats is the central aspect of digitization, which is the starting point for all modern automation solutions. The communication between air traffic controllers (ATCos) and pilots has been excluded so far, but the content of this communication is of significant importance for the automation systems. The verbal clearances are the main direct output of ATCos' work. Currently only the radar data – an indirect and delayed consequence of the ATCos' work – is considered by succeeding ATM functionalities. Additionally, using the ATCo-pilot communication utterances enables the digital world to improve situation monitoring, decision making, and post-operations' analysis processes. We assume that direct radio communication between ATCos and pilots will continue in the coming years, despite the upcoming use of data links such as controller pilot data link communications (CPDLC). Furthermore, it can also be assumed that both communication methods will exist in parallel for a considerable time during the transition phase. This means that ATCos still have to enter the content of their spoken commands into the ATC system via keyboard/mouse if no automatic digitization solution exists.

Recently, Assistant Based Speech Recognition (ABSR), which originated from the AcListant® project [3], has shown to be a vital component in the digitization of verbal ATCo commands. In general, ABSR integrates ASR with an Assistant System to generate the situational context, which is based on surveillance data (e.g., radar data, weather information) [4]. The provided context allows ABSR to generate a reduced search space for the ASR engine that fits to the current air traffic situation. Additionally, it also enables to check and modify the results of *normal ASR*. The AcListant® and its successor project AcListant®-Strips have validated that ABSR reduces controllers' mouse clicking time by a factor of three [5] and reduces fuel consumption by 60 liters per flight [6]. However, the adaptation of ABSR to different airports or ATCo positions (e.g., tower, departure) is time and budget consuming. The adaptation requires a lot of manual adjustments to the selected environment. It also requires collecting large amounts of data and expert knowledge to carry out the adaptations.

The Horizon 2020 SESAR funded project MALORCA (Machine Learning of Speech Recognition Models for Controller Assistance) [7] chooses machine learning (ML) as a potential solution to provide a generic, cheap, and effective way for the adaptation process. The solution makes use of the radar data and voice recordings that are generated every day in the air traffic control operation rooms around the world. This data is used as input for machine learning algorithms that automatically learn and adapt the necessary components of the ABSR system. The MALORCA project proposes conceptual models, which consist of building blocks and airport dependent models.

Fig. 1 illustrates the information flow between the building blocks (marked in blue), automatically trained models (marked in green), and the manually adapted models (marked in orange). The building blocks are active processes that fulfil certain tasks. They are generic so that they can be reused for different controller positions and environments. Only the data and models are specific for each environment and need to be adapted or recreated through machine learning algorithms. The building blocks

and their effects on recognition performance were first presented in [8].

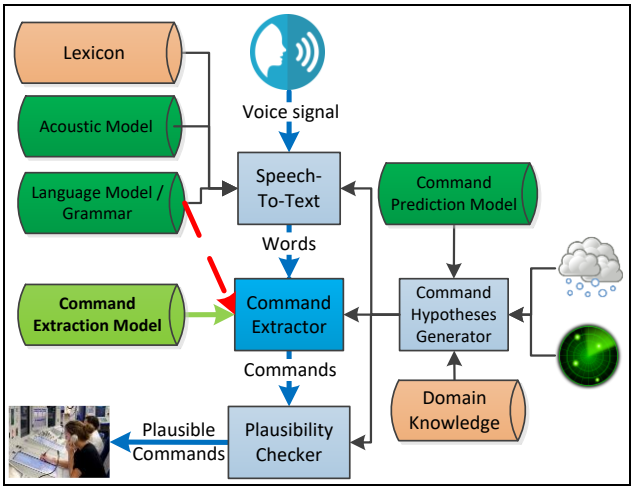

Fig. 1. Building Blocks and Models of an ABSR system.

As shown in Fig. 1 MALORCA proposes that the voice signal is transformed by the *Speech-To-Text* block (**S2T**) into a sequence of words (e.g., "*lufthansa one alfa bravo descend below four thousand feet*"). The *S2T* block is supported by the *Acoustic Model (AM)*, the *Lexicon*, and the *Language Model (LM)*. The *Lexicon* contains the used words and how they are combined by phonemes. The AM covers regional differences of spoken English, i.e., accents and dialects. Usually, Deep Neural Networks (DNN) are used for acoustic modelling, i.e., non-symbolic learning approaches (see sect. II.B). The *LM* defines the most probable sequence of words. After *S2T* the *Command Extractor* transforms the word sequence into ATC concepts and commands, the so called annotation. The output of the *Command Extractor*, bases on an ontology, i.e., a set of rules, defined by the SESAR project PJ.16-04 [9]. For the above sequence of words we would get "*DLH1AB DESCEND 4000 ft BELOW*". The *Command Extractor* is also supported by the *Command Hypotheses Generator*, which predicts the set of commands, which are possible at any given time by using e.g., the current radar and weather information. The output of the *Command Extractor* is further verified by the *Plausibility Checker* before being finally displayed on the user's screen.

A. Problem of the MALORCA Architecture

In the MALORCA ABSR architecture [8] the *Command Extractor* is closely coupled with the *S2T* block by using the same *Language Model* and *Grammar* (red arrow in Fig. 1). This induces less flexibility and exchangeability. Automatic *Language Model* training is based on non-symbolic learning approaches, i.e., DNN, with all the known advantages and disadvantages for the "safety first" oriented ATM world (sect. II.B). The biggest problem though was the very effortful manual adaptation of the airport dependent grammar linked with the *Command Extraction Model*.

B. Proposed Solution for Command Extraction

Therefore, this paper focuses on an improved *Command Extractor* building block and a new *Command Extraction Model* highlighted in bright blue and bright green, respectively, in Fig. 1. Details are provided in section IV.

C. Problem of Airport Dependent Phraseology

As mentioned above, the command extraction is based on a set of rules defined by the SESAR project PJ.16-04 and agreed with 14 European partners [9]. The *Command Extractor* building block uses the defined ontology rules to extract commands by transforming the sequence of words into a format that other applications can work with. For this task the *Command Extractor* has to deal with the fact that different ATCos use different words to express the same thing: "*turn left heading 040*", "*turn heading left 040*", "*turn left 040*", "*fly heading 040*", but also "*fly to the left 040*" and "*proceed left on a 040 heading*", all result in the same: the ontology concept "HEADING 040 LEFT". Asking ATCos, which phraseology they are using, the answers are quite different at least from our experience from having performed many manual transcription work. So again manual unwanted adaptation work of the airport dependent *Command Extraction Model* might be necessary.

Adapted Commercial-Off-The-Shelf (COTS) speech recognition engines used in the speech recognition exercises of the SESAR 2020 funded solution PJ.16-04 have shown that the sole use of COTS engines is still not an option to get the needed command recognition rates of greater than 90% [10]. The exercises conducted by Thales, DLR and the air navigation service providers from Czech Republic, Austria, and Croatia have demonstrated that it is in principle possible to manually adapt the grammar of the used COTS speech recognition engine to the phraseology deviation of the ATCos. Nevertheless, this is a time consuming and costly process.

D. Proposed Solution to avoid Manual Adaptation Work

In order to efficiently cover all the phraseology deviations that ATCos are using, the rules to transform the sequence of words, into the SESAR ontology have to be learned automatically, i.e., the local versions of the *Command Extraction Model* should be created and adapted automatically as far as possible.

E. Paper Structure

This paper is organized as follows. Section II discusses some related work carried out in the area of command annotations, followed by section III which presents the ontology proposed by different European ATM stakeholders for extracting the semantics from controller utterances. Section IV discusses how the *Command Extractor* is separated from the *Speech-To-Text* block by using a *Command Extraction Model*. Here we also show how the dependency of the *Command Extractor* on the *Language Model* and *Grammar* is removed completely. In section V, we discuss how local deviations of ATCos can be automatically learned from command annotations and controller utterances. Here, we use a symbolic approach to avoid problems related to the unpredictable behavior of Neural Networks. Section VI presents the validation results, where more than 10,000 ATCo commands from different airports are used as validation data. The consequence of integrating local phraseology deviations as compared to using the extended ICAO phraseology is also evaluated. In the last section, we conclude and provide an outlook.

II. Related Work

Section II.B touches symbolic and non-symbolic learning approaches. Beforehand section II.A describes related work with respect to extraction of semantic concepts from controller utterances, i.e., the annotation task. Command annotation consists of two succeeding tasks, namely transcription and annotation. The **transcription** task involves the speech-to-text transformation, writing down word-by-word, what the ATCo has said. Examples are: "*lufthansa two bravo alfa descend flight level eight zero and reduce speed two two zero knots*" and "*bonjour air_france two seven three _aeh_ confirm vien* correction contact vienna radar on one two nine decimal five*". The **annotation** task extracts the semantic concepts from the transcriptions (text-to-concepts transformation), e.g., "*DLH2BA DESCEND 80 FL, DLH2BA REDUCE 220 kt*" and "*AFR273 CORRECTION, AFR273 CONTACT VIENNA_RADAR, AFR273 CONTACT_FREQUENCY 129.500*".

A. Command Annotation Ontologies

The SESAR funded activity PJ.16-04-ASR has shown that different European stakeholders have common objectives when integrating ASR into ATM applications, but their output formats are still very different. This triggered the project partners to agree on an ontology that consists of unique rules to transcribe and annotate commands [9] as presented in section III. PJ.16-04 partners agreeing on a common European-wide available ontology include Air Navigation Service Providers (ANS CR, Avinor, Austro Control, DFS, LFV, NATS, Romatsa), research institutes (CRIDA, DLR), ATM supplier industry (Frequentis, Indra, and Thales), and Integra as ATM consultancy. Before going into details, we present other approaches to modelling extracted concepts from ATC utterances.

Nguyen and Holone [11], [12] proposed 10 classes to replace word sequences with their corresponding class labels: 1) callsign, 2) unit-name, 3) fix, 4) number, 5) letter, 6) greeting, 7) non-verbal articulations (ah, yeah, aha, etc.) and the three minor classes 8) directions (left, right, etc.), 9) position (above, below …), 10) unit (feet etc.). Johnson et al. [13] propose a keyword and value representation in JSON format [14], where keywords could be Callsign, ToFix, FlightLevel, Altimeter, etc.

In the AcListanct® project [3], Saarland University and DLR created an ontology which consists of four elements: 1) callsign, 2) command type, 3) command value, and 4) unit [15], [16]. Callsign and command type are mandatory. Unit is only used for altitude commands, i.e., only flight level and feet are used. More than 30 command types were initially supported. This approach reached its limits in the MALORCA project [17], when it was extended for live traffic in Vienna and Prague approach. Additionally, departing and overflight traffic had to be taken into account. Therefore, an increasing number of command types (e.g., QNH, INFORMATION, REPORT_SPEED, EXPECT_RUNWAY) had to be supported, which also created the necessity to annotate conditional clearances [18].

In 2002, NATS conducted a four-year project to look at the possible applications of ASR within the London Terminal environment [19]. Initially, several ontologies were proposed based on a Statistical Language Model. At project closure, the ontology encompassed five elements: 1) callsign, 2) standard (International phraseology) type, 3) non-standard (i.e., NATS based procedures) type, 4) type value, and 5) type currency/unit (e.g., feet, degrees). However, air traffic controllers do not always strictly adhere to the standard phraseology. In 2011, it was found that a large volume of the surplus language used was not standard phraseology or required for flight information. One of the NATS terminal approach sectors found that more than 20% of the utterances consisted of additional information and courtesies. For example, the transmission "*speedbird two one maintain speed until advised any delay will be less than ten*" includes information that is outside the standard phraseology and is, therefore, challenging for an ASR service.

DFS started the introduction of Voice Recognition and Response for the DFS controller training to replace simulation pilots. The simulator integrates the BBN HARK AVOKE finite state speech recognition system. It successively started to implement the DFS standard phraseology [9]. Since 2008, ENAIRE, EML, and CRIDA have been working together on an ASR prototype called VOICE. The speech recognition module of the prototype produces a transcription of the audio segment, either in English or in Spanish. From the transcription, callsigns and events are identified according to the defined grammar rules, which are used to create the final XML output file. INDRA is continuously enhancing an autonomous speech recognizer. Currently, the recognized phraseology includes more than 50 words based on the ICAO's standard [9]. Since 2013, Thales has been working on a platform called 'Shape', which integrates various modalities that include an ASR module based on the Recognizer 10 engine from Nuance Communications. The voice recognition engine is assisted by a hypotheses engine developed by Thales. Given an audio segment and a grammar file generated by the hypotheses engine, the voice recognition engine produces a transcription of the audio segment using XML tagging [9].

B. Symbolic versus non-symbolic approaches

Machine Learning (ML) can be divided into different categories: Supervised versus unsupervised learning or symbol versus non-symbolic approaches. **Symbolic** approaches are based on machine as well as human readable knowledge, i.e., representation. These approaches were already dominant in Artificial Intelligence (AI) research from the 1950s until 1990s. Successful applications are expert systems with production rules. **Non-symbolic** approaches have a similarly long history dating back to the 1940s. The most prominent examples are Artificial Neural Networks (ANN). Non-symbolic approaches, which not only include ANN, but also deep learning and ML models such as support vector machines, Bayesian networks, and Hidden Markov models. Currently non-symbolic approaches initiated a new AI hype, which could be partly attributed to the hardware advances, e.g., using the computing power of the graphic processing units (GPU) [20].

In **supervised learning**, all available training data is labeled, whereas no labeled training data is available for **unsupervised learning**. Semi-supervised learning uses both labeled and a large amount of unlabeled data. A speech recognizer can either be treated as one big ML problem (i.e., we try to directly map the audio signal to the final ATC concepts) or it can be broken down into separate ML problems for the acoustic model (AM) and for the language model (LM). Recently semi-supervised learning of NN based AM for special domains like YouTube

videos has been performed [21]. For LM, neither unsupervised learning nor semi-supervised learning has been very successful. Only supervised learning has been an established technique. For adapting non-neural LMs, Bellegarda [22] gives a good overview. For adapting ANN in a supervised fashion, this can either be done in a rescoring step [23] or by mapping the NN to a tree directly in the first pass for speech recognition [24]. We split the LM into the part supporting the word extraction and a Command Extraction Model, which extracts the ATC concepts from the extracted word sequences. A symbolic approach with semi-supervised learning is used. Both symbolic and non-symbolic approaches have advantages and disadvantages. Non-symbolic approaches are more robust against noise, but require a lot of training data. As in our case enough domain knowledge is available, we can benefit from the better transparency of the symbolic reasoning. The results can easily be explained to human experts, due to explicit symbolic knowledge representation.

III. Ontology of PJ.16-04

A subset of the PJ.16-04 ontology for transcription and annotation is presented in the following. Transcription rules encompass the agreement to only use English letters (no "ä", no "é"), to write "*nine*", even if "*niner*" was said as defined in the ICAO phraseology, to use "*" if an uttered word was not spoken until the end, to not use capital letters, to use "*alfa*" and not "*alpha*" etc. Annotation rules for example encompass the agreement on using upper case letters for callsigns and command types, repeating the callsign for each separate command even if it was not repeated by the controller or using NO_CALLSIGN if not uttered at all, noting frequency with six digits separated by a point in the middle even if the controller uttered less digits [9].

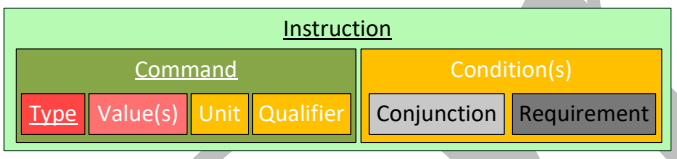

Fig. 2. *Elements of an Instruction for a Clearance* [9].

Each annotated command consists of a callsign and an instruction. Fig. 2 shows that an instruction always comprises of a command (darker green part is mandatory) and one or more optional (orange) conditions. A command is again composed of a type (or combined type). Depending on the type one or more values, units such as *FL*, *ft* or *none*, and qualifiers such as *LEFT, RIGHT, OR_LESS, BELOW* must or may follow. An utterance may consist of multiple instructions for the same callsign or even for different callsigns. After the ontology was first presented in [9], many improvements were integrated, e.g., new command types. The main improvements, however, include many examples, how special utterances should be converted into ATC concepts according to the ontology. Furthermore, it was agreed that it is outside the ontology how to map certain word sequences ("bravo", "taxiway alfa", "apron") to ATC concepts. Hence, the example "roger via bravo taxiway alfa to the apron" can result in "NO_CALLSIGN TAXI VIA B A, NO_CALLSIGN TAXI TO APRON" or in "NO_CALLSIGN TAXI VIA TXW_B TXW_A, NO_CALLSIGN TAXI TO APRN". The ontology definition currently comprises 116 different command types with 42 parameters that cover en-route, approach, tower, and ground controller commands as well as pilot requests. This ontology is currently used and enhanced in the SESAR2020 wave-2 industrial research projects PJ.05-97 and PJ.10-96, as well as the exploratory research project HAAWAII, and the German national funded project STARFiSH.

Although having now the ontology rules for transforming a word sequence into ATC concepts, it is still a challenge that two different persons transform a sequence of words into the same ATC concepts. This is a prerequisite to compare the results of different speech recognition engines. Helmke et al. analyzed the recognition and rejection errors of their ASR system. It turned out that the output of the ASR system (transcription and annotation) often was correct, but the corresponding manual annotation, i.e., validation data, was just wrong [25]. Therefore, we are currently implementing a new *Command Extractor* (Fig. 1), which automatically transforms ATC utterances into the corresponding ATC concepts. The symbolic approach, presented in the next section, even enables to automatically learn these annotation rules.

IV. Command Extraction Algorithm

The proposed *Command Extraction Model* tries to identify patterns occurring in ATCo utterances in order to extract controller commands. This consists of three steps on a broader perspective, which are callsign extraction, command extractions using keyword sequences, and extraction of commands from known ATC Concepts and unrecognized words from the utterance. The following Fig. 3 illustrates the complete command extraction algorithm:

```
Extract Predicted Callsign from initial words;
while (NOT end of utterance reached)
{
    Extract command when matching keyword sequence found;
    if (value, unit, qualifier ... could be extracted)
    {
      Add command to extracted commands;
    }
}
while (NOT end of utterance reached)
{
    Find ATC concept in unmatched words;
    if (unit, qualifier ... could be extracted)
    {
       Add command to extracted commands;
    }
    Find command hints in unmatched words;
    if (value, unit, qualifier ... could be extracted)
    {
       Add command to extracted commands;
    }
}
while (NOT end of utterance reached)
{
    Find first or further callsigns in unmatched words;
}
```

Fig. 3. *Callsign and Command Extraction Algorithm*.

The used algorithm could be best explained by using an example of a controller utterance: "*good morning austrian two tango descend nine zero or below ils three four speed up to you*

expect a straight in". We use an ABSR system that provides the set of commands which are feasible in the given situation. Therefore, the set of possible callsigns at any given time is also available, e.g., *AUA1AC6, AUA472T,* and *NLY3618* (predicted Callsigns). We first try to extract the callsign from the initial words of the utterance as illustrated in line 1 of the algorithm (Fig. 3). In the given example, "*austrian two tango*" results in the extraction of the callsign *AUA472T*. Next, we try to find keywords in the utterance, which match a command type. Keywords such as "*reduce speed*", "*reduce your speed*", "*start reducing*", "*speed reduction*" etc. may start a *REDUCE* command, while keywords "*no speed restrictions*", "*free speed*", "*speed up to you*" etc. may start a *NO_SPEED_RESTRICTIONS* command. Similarly, using the keywords *"descend", "confirm descending", "continue down"* defined for *DESCEND*, we identify that we may have a *DESCEND* command when we encounter *"descend"* in the utterance. The ontology defines that *DESCEND* commands must always have a *value* and may have optional *unit, qualifier* or *condition* fields. We try to extract them before and after the keyword "*descend*" in the utterance. If the extraction of the mandatory *value* for DESCEND fails, the next matching keyword sequence is processed. In the given case, however, we successfully extract the DESCEND command resulting in "AUA472T DESCEND 90 none OR_BELOW". The command extraction is illustrated between lines 2 and 9 in Fig. 3.

Similarly (still lines 2 and 9 in Fig. 3) we extract "AUA472T NO_SPEED_ RESTRICTIONS" by using the next matching keyword sequence *"speed up to you"* from the utterance. The given example can now be color-coded and rewritten as: "*good morning austrian two tango descend nine zero or below ils three four speed up to you expect a straight in*". Here, colors *red*, *green*, and *blue* mark the extracted callsign, *DESCEND,* and *NO_SPEED_RESTRICTIONS* commands, respectively. The words displayed in black are the ones, which have not yet been used for command extraction. Although these words do not match any keyword sequences, they can still be used to extract commands by identifying known ATC concepts or unmatched words, which is done in the next step as illustrated between lines 10 and 22 of the algorithm in Fig. 3. ATC concepts include waypoints, runways, taxiways, frequencies, etc. In the given example, "*three four*" represents a runway, namely *34*. Together with the unmapped words before and after the concept name, we successfully extract "*AUA472T EXPECT ILS 34*". We further examine the remaining unmatched words in the utterance for indications regarding command types. For example, "knots" could suggest either *SPEED* or *INFORMATION WINDSPEED* commands. Similarly, "*feet*" and "*decimal*" could suggest *ALTITUDE* and *CONTACT_FREQUENCY* commands, respectively. In addition, fillers such as greetings (e.g., *"good morning"*) are also recognized.

The command extraction is followed by a second round of callsign extraction in order to extract callsign(s) which were not previously extracted as illustrated in lines 23 to 26 of the algorithm in Fig. 3. Here we apply Levenshtein distance calculations [26] to evaluate the best matching callsign. Hence, callsign AUA472T is identified as the best match even for "austrian seven three tango". The second callsign extraction step also determines whether we have different callsigns in the same utterance given by the "*break break*" command. The reason for carrying out the second round of callsign extraction after extracting all commands is to overcome the challenge of mixing up of command and callsign symbols leading to wrong callsign/commands getting extracted. For example, in "*lufthansa alfa one charlie papa tower descend* …", the word "*papa*" could either be a letter from the callsign or could specify the Hungarian air base Pápa. Carrying out callsign extraction for unrecognized callsigns at the end would ensure that "*papa*" is correctly associated with the respective callsign/command, in this case with the *STATION* command.

V. Learning from Transcription-Annotation Pairs

In addition to extracting commands, the proposed solution also learns possible keyword sequences, ATC concepts, and airline designators from transcription-annotation pairs. In this section, we define the problems and give a brief introduction on our solution.

For example, using the defined ontology the annotation for the utterance "*goodbye sky_travel three bravo charlie climb flight level seven zero and proceed direct oscar papa four zero five bye*" results in the commands *"TVS3BC CLIMB 70 FL" and "TVS3BC DIRECT_TO OP405 none"*. Let us assume that the proposed *Command Extraction Model* extracts the *CLIMB* command, but not the *DIRECT_TO* command. In order to aid the extraction process, we learn from the corresponding annotated command *("TVS3BC DIRECT_TO OP405 none*") which was not extracted. We also assume that *"goodbye"* and *"bye"* are classified as *"greeting"*, *"sky_travel three bravo charlie"* as *"callsign"*, *"climb flight level seven zero"* with *"type, unit, unit, value, value"* of first command and all the other words, i.e., *"and proceed direct oscar papa four zero five" as "unknown"*. We intend to learn new keyword sequences, concepts, and airline designators from these *"unknown"* words.

A. Learning of ATC Concepts

For learning concepts, we assume that keyword sequence *"proceed direct"* initiates the extraction of DIRECT_TO command, which is followed by a waypoint value. Therefore, the word sequences *"oscar", "oscar papa", …, "oscar papa four zero five", …, "zero five"* and *"five"* form candidates for the waypoint concept OP405. When multiple examples for the ATC concept OP405 are available, then finally the word sequence *"oscar papa four zero five"* is learnt for OP405.

B. Learning of new Phraseology

Similarly, phraseology deviations can also be learnt from transcription-annotation pairs. For the given example, we intend to learn "*proceed direct*" as a phraseology keyword sequence for the DIRECT_TO command. We assume that "*oscar papa four zero five*" is a known keyword sequence for waypoint OP405. We learn word sequences "*and*", "*and proceed*", "*and proceed direct*", "*proceed direct*", "*direct*" as candidates for DIRECT_TO keywords, resulting in "*proceed direct*" if many transcription-annotations pairs are available. Learning is an iterative process. When we start command extraction for a new airport using limited phraseology sequences, some new keyword sequences and ATC concepts are learned. While implementing the learned keywords and concepts, some more new keyword

sequences and concepts are learned. Hence, the learning keeps evolving throughout the process.

C. Learning of Airline Designators

Airline designators are also learnt in the same way. *"airest cargo"* is the official word sequence for the three letter airline designator "AEG". Nevertheless, based on our data, *"airest"*, *"eastern"* and *"east air"* are also used by ATCos. From the word sequence *"hello east air four hundred alfa climb one one zero"* we have the ontology annotation "AEG400A CLIMB 110 none". If only "*four hundred alfa*" is marked as belonging to the callsign, then none of these words correspond to the airline designator "AEG". We, therefore, learn the sequences *"east"*, *"east air"* and *"air"*, which form the candidates for the designator AEG from which "east air" is learnt. Since *"hello"* is a greeting here, it is not considered as a candidate for the designator AEG.

VI. Validation Trials and Results

The described command extraction and command extraction model learning approach is validated on 12,979 commands taken from Prague approach ATCo-pilot conversation. Hence, audio data contained English language with Czech accent. We first show the extraction accuracy on manually annotated data, when we manually create the *Command Extraction Model*. Section VI.B shows the performance on the output of different Speech-to-Text implementations, whereas section VI.C shows the results when we automatically learn the ATC concepts of the *Command Extraction Model*. Subsection VI.D shows, how important it is, to model local ICAO phraseology deviations and to extract from incomplete utterances.

A. Extraction Accuracy

Table 1 shows the extraction accuracy results. 6,885 commands were taken from simulation trials with 5 ATCos [8] and 6,094 from operational environment recordings of 12 ATCos [10]. The 12,979 commands resulted from 7,249 utterances (4,211 from simulation trials and 3,038 from the operational environment). The number of commands per utterance was between one and seven.

Table 1: Extraction Accuracy for Ops Room and Lab Data

	# Cmds	# Utterances	ExtR	ErrR	CaExRa
Ops Room	6094	3038	98.47%	0.92%	99.77%
Lab Data	6885	4211	99.17%	0.51%	99.67%
All	12979	7249	98.84%	0.70%	99.71%

The extraction rate (ExtR) is the number of correctly recognized commands divided by the total number of commands. A command is correctly recognized only, if the callsign, the type including second type, the value, the unit, the qualifier, and the condition are all correct. So "DLH123 SPEED 220 none" is different from "DLH123 SPEED 220 kt". Then we have an error contributing to the error rate (ErR). The difference to 100% is the rejection rate. A wrong extraction is counted only as a rejection, if it is not shown to the controller, i.e., (i) nothing is extracted for a given command or (ii) type NO_CONCEPT (iii) or callsign NO_CALLSIGN is extracted and this differs from the given command. Table 1 also shows the callsign extraction rate (CaExRa), when the callsign is extracted correctly. ABSR predicts the commands, which are possible in the current situation. The accuracy of the predictor was approximately 99% based on the evaluations of [8] and [10].

We investigated the reasons why we could not obtain 100% correct extractions. A rate of even 98.8% still means that more than 100 commands of the given data set are not recognized as annotated. 50% of them still result from annotation errors. Other problems are e.g., (i) the four words "one one eight one" are not extracted as "NO_CALLSIGN CONTACT_FREQUENCY 118.100", (ii) "that's correct one six zero" is not extracted as "NO_CALLSIGN ALTITUDE 160 none", (iii) usage of not existing callsigns or (iv) problems with the "correction" phraseology.

B. Recognition Performance

On the other hand, it is not important for ASR applications that the extraction from manual transcriptions is good, but a good extraction from the output of the Speech-To-Text block is required. For evaluating the performance of the approach on automatic transcriptions, we use the output of an ABSR system, which was already used in [8] to evaluate the contribution of the different building blocks of an ABSR system, i.e., we only show the performance on the ops room data. Row "No Callsign Information" in Table 2 shows the results in case that the Speech-To-Text block gets no information, which callsigns are in the air. Row "Callsign Information" shows the results if the callsign information is provided. Row "Female Speaker" shows the results, when callsign information is provided, but the acoustic model is just trained with the training data from just one female speaker. ExtR and CaExRa show again command extraction rate and callsign extraction rate. WER is the word error rate, i.e., it corresponds to the quality of the Speech-To-Text block.

Table 2: Extraction Performance on Speech-to-Text Outputs

	ExtR	CaExRa	WER
No Callsign Information	96.5%	98.7%	2.3%
Callsign Information	96.6%	98.2%	2.8%
Female Speaker	76.8%	88.5%	13.5%

Table 2 shows that the extraction rate of Table 1 degrades from 98.5% to 96.5% or 96.6%, if we have a good word recognizer. When the performance of the word recognizer is bad (WER > 10%), the command extraction performance is also poor. Surprisingly, the WER goes down when callsign information is available. The reason is that the Speech-To-Text block tries to "press" the voice signal into allowed expected word sequences. Here, it is better to extract the callsign from the word sequence afterwards (WER in Table 1 of course is 0%).

C. Learning Performance

The used ATC concept database consists of 59 different concepts (runway, waypoint, frequency position names). They were all deleted and automatically learned as described in Section V. Row "Before Learning" in Table 3 shows the extraction results without any modelled ATC concept (column "# Concepts"). The bad extraction rate of 72.1% results from the fact that no *DIRECT_TO* or *CLEARED ILS* types are recognized. After the first iteration we already learned 52 concepts resulting in a big improvement of the extraction performance. The second iteration only adds one new waypoint, but five additional word sequences. A third iteration provides no improvements. With only

some manual corrections of the word sequences, used to extract an ATC concept, we nearly achieve the original extraction performance of Table 1. The word sequence "*oscar romeo four zero five gedern next*" is e.g., manually deleted as valid word sequence for the waypoint "OR405". This was an abbreviation for "*oscar romeo four zero five and the next waypoint is gedern",* i.e., *gedern* is a waypoint name.

TABLE 3: EXTRACTION PERFORMANCE DEPENDING ON LEARNING

	ExtR	ErrR	# Concepts
Before Learning	72.06%	4.41%	0
First Iteration	94.56%	1.83%	52
Second Iteration	98.64%	1.12%	53
Manual correction	98.77%	0.72%	53
Original of Table 1	98.84%	0.70%	59

The previous experiment used the same data for training and for testing. This is of course not permitted. Therefore, we then only used 80%, 60%, etc. of the data for training. The results after the first iteration are presented in Table 4. With only 40% of the data we are already approaching the final extraction rate. However, we have learned only 35 of the 59 ATC concepts. Missing are only some rarely used waypoint names. Therefore, we also show the extraction rate for the *DIRECT_TO* command type, for which we get different results depending on training data size.

TABLE 4: LEARNING PERFORMANCE WITH ONLY A SUBSET OF TRAINING DATA

Training data Size	ExtR	ErrR	# Concepts	DIRECT_TO
20%	94.2%	1.9%	28	84.1%
40%	94.4%	1.8%	35	88.8%
60%	94.5%	1.8%	43	90.4%
80%	94.5%	1.8%	47	90.3%
100%	94.6%	1.8%	52	91.3%

D. Phraseology deviations

The last experiment answers the question of how important it is to model phraseology deviations in the *Command Extractor*. We divide our results (extraction rate and error rate) based on the data i.e., Ops Room data and Lab data. On modelling all major and minor deviations of the phraseology, we obtain extraction rates of 98.46% and 99.19% in the Ops Room and Lab environment, respectively. We next identify and delete all the big deviations in the phraseology. Here, all keyword sequences which deviate from the standard phraseology by a major degree were identified by an ATCo. An example for such major deviations includes, saying *"no speed"* instead of *"no speed restrictions"*.

TABLE 5: STANDARD PHRASEOLOGY AND ALLOWING DEVIATIONS

Experiment	Ops Room Data		Lab Data	
	ExtR	ErrR	ExtR	ErrR
Original	98.46%	0.98%	99.19%	0.51%
Big Deviations	98.43%	0.92%	99.19%	0.51%
Only one digit numbers	98.11%	0.93%	99.14%	0.55%
Standard Phraseology	88.74%	5.90%	86.10%	8.64%
Incomplete Commands	78.57%	0.85%	79.23%	0.49%

Table 5 shows that the extraction rates are almost the same, which means that ignoring the big deviations does not affect the extraction results. Next, on allowing only single digit numbers (i.e., numbers from zero to nine) as well as hundreds and thousands (which account for altitude values), the extraction rates decrease marginally by around 0.3% and 0.05% for Ops Room and Lab data, respectively. On further restricting the keyword sequences to the standard phraseology, the extraction rates drop significantly by around 10-13% for both environments. Here, keyword sequences such as *"start reducing speed"* and *"switch to"* are no longer accepted keyword sequences for REDUCE and CONTACT commands, respectively. Additionally, when incomplete commands are also not extracted (i.e., commands extracted from unmatched ATC concepts and words between lines 10 to 22 in Fig. 3), the extraction rates further decrease by 10%.

E. Validation Summary

We have shown that our approach is already able to achieve an extraction accuracy of nearly 99%. This is better than our achieved accuracy resulting from manual annotations by different persons. Performance on lab data is 0.5% higher than on real life data, which gives strong hints that ATCos behave differently in a simulation environment than in real life. We also tried our approach on transcribed data resulting from another airport and on voice recordings from remote tower experiments. We expect similar results as soon as the annotation errors resulting from wrong interpretation of the PJ.16-04 ontology are corrected.

The approach also performs very well on automatic transcriptions, i.e., outputs from the Speech-to-Text block of an ABSR system. The command recognition results are with 96% more than four percent better than the results of 92% reported in Table V in [27] on the same data, if we have good word recognition rates. With bad word recognition rates we received only similar performance as the result in [27]. We also showed that manual creation of the *Command Extraction Model* is not necessary. The model can be automatically trained from transcription-annotation pairs by using a symbolic rule-based approach. The symbolic approach requires less training data. 20% to 40% of the available data already resulted in nearly the same performance as we get using 100% of the data. Furthermore, the results of the used rule-base symbolic approach are easily explainable to human experts and to the ATCos increasing their thrust into the system. Slight deviations of the ICAO phraseology and especially incomplete and abbreviated utterances must be recognized. Otherwise the recognition performance degrades by more than 20% absolute resulting in ATCo's frustration when using ABSR output.

VII. CONCLUSIONS AND OUTLOOK

ABSR has proven to enable valuable support for ATCos in terms of workload reduction if commands are automatically digitized. However, it is hard to evaluate the ABSR performance by comparing target data with actual data if the target data is wrongly or inconsistently annotated by different persons. Here, the target data is the manual annotation of ATCo command contents considering the transcription of the ATCo utterance. An ontology harmonizes the expected target output, but still there is a chance to include manual errors. Hence, a checker module should automatically analyze annotations for common errors. Nevertheless, there will still remain many uncommon errors in the annotations. When transforming the transcription automatically into the annotation, this command extraction process

reduces the uncommon errors and supports to receive more accurate target data. Otherwise comparison of different ATC speech recognition engines is very difficult or impossible. The presented automatic annotation helps the speech understanding and is even learnable just from accurate target annotations and the corresponding transcriptions. We have shown that machine learning results in roughly the same accuracy level of target data as the manual approach, but it is much faster. Our basic *Command Extraction Model* only extracts 60% of ATCo commands. If the extraction model learns automatically, 98% of commands are covered. Therefore, it is easy to adapt the extraction process from one environment such as an ATCo working at a certain airport to another. Furthermore, the automatic command extraction enables functionalities like the analysis of ATCo utterances with respect to phraseology conformity. In addition, any ASR engine can be used to deliver the transcription as the command extraction for annotation is now an encapsulated further processing step. The command extraction module will further be enhanced and adapted for the ATC tower environment.

ACKNOWLEDGMENT

Sub-parts of the DLR Command Extraction module have been developed in various projects on Assistant Based Speech Recognition. Three SESAR2020 industrial research projects PJ.16-04 CWP HMI (Wave-1), PJ.10-96 (Wave-2), and PJ.05-97 (Wave-2), all comprising an automatic speech recognition activity, and the exploratory research project HAAWAII have received funding from the SESAR Joint Undertaking under the European Union's grant agreement No. 734141, 874464, 874470, 884287 as well as the project STARFiSH funded by the German Federal Ministry of Education and Research.

REFERENCES

[1] SESAR homepage, https://www.sesarju.eu, no date (n.d.).

[2] FAA, "Modernization of U.S. airspace," https://www.faa.gov/nextgen/ n.d.

[3] AcListant homepage: www.AcListant.de, AcListant = Active Listening Assistant, n.d

[4] H. Helmke, J. Rataj, T. Mühlhausen, O. Ohneiser, H. Ehr, M. Kleinert, Y. Oualil, and M. Schulder, "Assistant-based speech recognition for ATM applications," 11th USA/Europe Air Traffic Management Research and Development Seminar (ATM2015), Lisbon, Portugal, 2015.

[5] H. Helmke, O. Ohneiser, T. Mühlhausen, and M. Wies, "Reducing controller workload with automatic speech recognition," IEEE/AIAA 35th Digital Avionics Systems Conference (DASC), Sacramento, California, USA, 2016.

[6] H. Helmke, O. Ohneiser, J. Buxbaum, and C. Kern, "Increasing ATM efficiency with assistant-based speech recognition," 12th USA/Europe Air Traffic Management Research and Development Seminar (ATM2017), Seattle, Washington, USA, 2017.

[7] M. Kleinert, H. Helmke, G. Siol, H. Ehr, M. Finke, and A. Srinivasamurthy, "Machine learning of controller command prediction models from recorded radar data and controller speech utterances," 7th SESAR Innovation Days, Belgrade, 2017.

[8] M. Kleinert, H. Helmke, H. Ehr, Chr. Kern, D. Klakow, P. Motlicek, M. Singh, and G. Siol, "Building Blocks of Assistant Based Speech Recognition for Air Traffic Management Applications," 8th SESAR Innovation Days, Salzburg, Austria, 2018.

[9] H. Helmke, M. Slotty, M. Poiger, D.F. Herrer, O. Ohneiser et al., "Ontology for transcription of ATC speech commands of SESAR 2020 solution PJ.16-04," IEEE/AIAA 37th Digital Avionics Systems Conference (DASC), London, United Kingdom, 2018.

[10] M. Kleinert, H. Helmke, S. Moos, P. Hlousek, C. Windisch, O. Ohneiser, H. Ehr, and A. Labreuil, "Reducing Controller Workload by Automatic Speech Recognition Assisted Radar Label Maintenance," 9th SESAR Innovation Days, Athens, Greece, 2019.

[11] V.N. Nguyen and H. Holone, "N-best list re-ranking using syntactic score: A solution for improving speech recognition accuracy in Air Traffic Control," 16th Int. Conf. on Control, Automation and Systems (ICCAS 2016), Gyeongju, Korea, 2016, pp. 1309–1314.

[12] V.N. Nguyen and H. Holone, "N-best list re-ranking using syntactic relatedness and syntactic score: An approach for improving speech recognition accuracy in Air Traffic Control," 16th Int. Conf. on Control, Automation and Systems (ICCAS 2016), Gyeongju, Korea, 2016, pp. 1315–1319.

[13] D.R. Johnson, V.I. Nenov, and G. Espinoza, "Automatic speech semantic recognition and verification in Air Traffic Control," IEEE/AIAA, 32rd Digital Avionics Systems Conference (DASC), East Syracuse, NY, USA, 2016.

[14] http://www.json.org, JSON = JavaScript Object Notation, n.d.

[15] A. Schmidt, "Integrating situational context information into an online ASR system for Air Traffic Control," Master Thesis, Saarland University (UdS), 2014.

[16] Y. Oualil, M. Schulder, H. Helmke, A. Schmidt, and D. Klakow, "Real-time integration of dynamic context information for improving automatic speech recognition," Interspeech, Dresden, Germany, 2015.

[17] MALORCA homepage: www.malorca-project.de, MALORCA = Machine Learning of Recognition Models for Controller Assistance, n.d.

[18] A. Srinivasamurthy, P. Motlicek, I. Himawan, G. Szaszák, Y. Oualil, and H. Helmke, "Semisupervised learning with semantic knowledge extraction for improved speech recognition in air traffic control," INTERSPEECH 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden Aug. 2017.

[19] D. Randall, "Direct Voice Input (DVI) Technology readiness and status introduction," Whitely, Fareham, UK, 2006.

[20] B.G. Humm, H. Bense, M. Classen, St. Geißler, Th. Hoppe O. Juwig et al.: "Current trends in applied machine intelligence," Informatik_Spektrum Vol. 42, No. 1, 2019, pp 28-37.

[21] H. Liao, E. McDermott, and A. Senior, "Large scale deep neural network acoustic modeling with semi-supervised training data for YouTube video transcription", IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, 2013, pp. 368-373.

[22] J. Bellegarda, "Statistical language model adaptation: Review and perspectives", Speech Communication, 2004.

[23] X. Chen, T. Tan, X. Liu, P. Lanchantin, M. Wan, M.J.F. Gales, and P. Woodland, "Recurrent Neural Network Language Model Adaptation For Multi-Genre Broadcast Speech Recognition", ISCA Interspeech, Dresden, 2015.

[24] M .Singh, Y. Oualil, and D. Klakow, "Approximated and domain-adapted LSTM language models for first-pass decoding in speech recognition", in Proceedings of the 18th Annual Conference of the International Speech Communication Association (INTERSPEECH), Stockholm, Sweden, September 2017, pp. 2720-2724.

[25] H. Helmke, M. Kleinert, J. Rataj, P. Motlicek, D. Klakow, C. Kern, and P. Hlousek, "Cost Reductions Enabled by Machine Learning in ATM -- How can Automatic Speech Recognition enrich human operators' performance?,", in 13th USA/Europe Air Traffic Management Research and Development Seminar (ATM2019), Vienna, Austria, 2019.

[26] V.I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in: Soviet Physics -- Doklady 10.8, Feb. 1966.

[27] M. Kleinert, H. Helmke, G. Siol, H. Ehr, A. Cerna, C. Kern, D. Klakow, P. Motlicek et al., "Semi-supervised Adaptation of Assistant Based Speech Recognition Models for different Approach Areas", in IEEE/AIAA 37th Digital Avionics Systems Conference (DASC). London, United Kingdom, 2018.