Machine Learning of Controller Command Prediction Models from Recorded Radar Data and Controller Speech Utterances

Matthias Kleinert[1], Hartmut Helmke[1],
Gerald Siol[1], Heiko Ehr[1], Michael Finke[1],
Youssef Oualil[2], Ajay Srinivasamurthy[3]
[1] Institute of Flight Guidance, German Aerospace Center (DLR), Braunschweig Germany,

[2] Spoken Language Systems Group (LSV), Saarland University (UdS), Saarbrücken, Germany
[3] Idiap Research Institute, Martigny, Switzerland

firstname.lastname@{dlr.de, lsv.uni-saarland.de, idiap.ch}

***Abstract*—Recently, the project AcListant® related to automatic speech recognition has achieved command recognition error rates below 1.7% based on Assistant Based Speech Recognition (ABSR). One main issue to transfer ABSR from the laboratory to the ops-rooms is its costs of deployment. Currently each ABSR model must manually be adapted to the local environment due to e.g. different accents and models to predict possible controller commands. The Horizon 2020 funded project MALORCA (Machine Learning of Speech Recognition Models for Controller Assistance) proposes a general, cheap and effective solution to automate this re-learning, adaptation and customization process to new environments, by taking advantage of the large amount of speech data available in the ATM world. This paper presents an algorithm which automatically learns a model to predict controller commands from recorded untranscribed controller utterances and the corresponding radar data. The trained model is validated against transcribed controller commands for Vienna and Prague approach. Command error rates are reduced from 4.1% to 0.9% for Prague approach and from 10.9% to 2.0% for Vienna.**

Keywords-Machine Learning, Assistant Based Speech Recognition, Unsupervised Learning, Command Prediction Model

I. INTRODUCTION

A. Problem

One of the main causes hampering the introduction of higher levels of automation in the Air Traffic Management (ATM) world is the cost factor. ATM system suppliers try to reduce costs by developing generic systems, e.g. one basic Arrival Manager like MAESTRO [1] which fits for many airports. Therefore, the deployment of decision and negotiation support tools in current ATM business still requires a strong and manual adaptation to the local environment to avoid low end-user (controller) acceptance. Every single process of adaptation yields a significant cost increase for a core ATM system so that total system costs easily exceed the threshold of one million Euros.

To ensure the acceptance of any new feature developed by any ATM project, it is imperative that its benefits are clearly recognizable for the end-user at the very beginning when they are confronted with new tools. If the outcomes of the development are not of certain quality, new problems or additional workload are introduced to the end-user and they will likely refuse the acceptance and cooperation, i.e. when a new tool is demonstrated to end-users it should already be tailored to the local environment. On the other hand if we provide end-users with innovative and helpful tools with a promising perspective that further improvements can be achieved, strong support from the end-users can be expected for the further development.

In aviation, Automatic Speech Recognition (ASR) is a known technology used with considerable success in ATC training simulators. Recently, the venture capital funded project AcListant® [2] has achieved command error rates below 2% in operational environments and fuel reduction of 60 liters per aircraft (based on an A320) with a specific speech recognition technology called Assistant Based Speech Recognition (ABSR), developed by Saarland University (USAAR) and DLR [3]-[6]. ABSR combines ASR with an assistant system, which generates context information to reduce the search space of the speech recognizer.

To manually adapt the ABSR models to different local environments, large amounts of speech data need to be collected and manually transcribed, especially, if the mismatch between the original models and the new environment is large. It requires significant human effort and the process of re-training requires vast expertise. The new data is generally used to adapt the acoustic model, the language model, and the rules to generate the context for the local environment. Moreover, partial adaptation may be required also during the life time of the system given that involved controllers, waypoints, procedures or used phraseology, etc. may change, resulting in new costs. Today the adaptation process requires a notable amount of resources and hence causes considerable costs. The estimate is that these adaptations costs represent at least 10% of the total costs [7]. Both end-user partners, i.e. Austro Control and ANS CR, are the first test sites who will explore proposed solutions. To conclude: Less expensive adaptation to the local needs without compromises with respect to end-user acceptance is needed.

The work is partly funded by SESAR Joint Undertaking (Grant Number 698824).

B. *Solution*

The Horizon 2020 SESAR project MALORCA (**Ma**chine **L**earning **o**f Speech **R**ecognition Models for **C**ontroller **A**ssistance) proposes a general, cheap and effective solution to automate this adaptation and customisation process. Adaptation of speech recognition models were selected as a first show-case of MALORCA [8]. The MALORCA consortium consists of two members from academia, Saarland University (Germany) and Idiap Research Institute (Switzerland), Air Navigation Service Providers from Czech Republic (ANS CR) and Austria (Austro Control) representing the user needs, and the German Aerospace Center (DLR) as the connecting element between fundamental research and business needs. The proposed solution builds on the huge amount of target data recorded every day in the operation rooms.

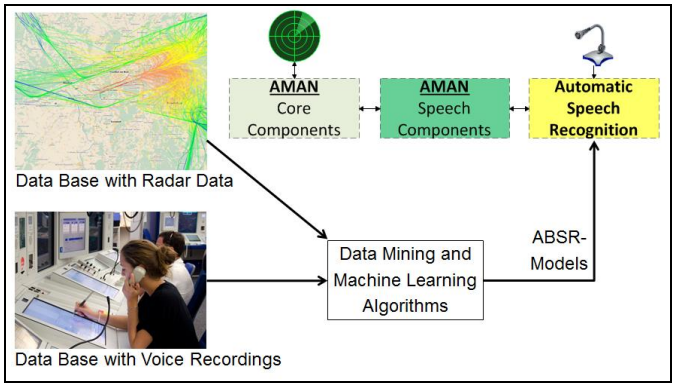

Figure 1 Principal idea of MALORCA project to learn from historic data

As shown in Figure 1, each Air Navigation Service Provider (ANSP) generates Mega Bytes or even Giga Bytes of radar data and voice recordings on a daily basis. These recordings can be the input for machine learning algorithms. The outputs would be improved speech recognition models which are adapted to the local needs. These initial models can be improved day by day. If a new waypoint is added it would be learned, if a waypoint is removed it would be "unlearned" etc.

C. *Paper Structure*

In the next chapter we present related work with respect to machine learning and speech recognition applications in ATM. In chapter III the Command Prediction Model which is the focus of this paper is shown in detail. The performed proof-of-concept experiments are described in chapter IV. The results can be found in chapter V. Afterwards results and next steps are discussed in chapter VI before we draw our conclusions in chapter VII. First results on learning the acoustic and language models can be found in [9] and [10].

II. Related Work

A. *Speech Recognition Applications in ATM*

Speech Recognition applications have dramatically improved during the last decade (e.g. Siri®, Alexa, Google Assistant). The integration of ASR in ATC training started already in the late 80s [11]. Today ASR applications go beyond simulation and training. ASR is e.g. used to get more objective feedback of controllers' workload [12]. Chen and Kopald used speech recognition to build a safety net for airport surface traffic to avoid aircraft using a closed runway [13]. Most recently they presented an approach to detect pilot read back errors [14].

Although the vocabulary in controller pilot communication is quite limited and phraseology is restricted, recognition rates are still far from being perfect. One promising approach to improve ASR performance is using context knowledge regarding expected utterances. These attempts go back to the 80s [15], [16]. This information may heavily reduce the search space and lead to fewer misrecognitions [17]. The approach developed by DLR and Saarland University uses the output of an Arrival Manager (AMAN) as context information [3]. The AMAN (4D-CARMA) in Figure 2 analyzes the current situation of the airspace and predicts possible future states used by the "Hypotheses Generator" to predict a set of possible commands. This dramatically reduces the search of the "Lattice Generator" [18], [19]. The search lattice is dynamically regenerated and contains a search tree for all possible phoneme sequences determined by the "Hypotheses Generator". The "Speech Recognizer" finds the most probable path in the search tree. The output of the "Command Extractor" is checked again by the "Plausibility Checker", determining whether the recognized commands are reasonable in the current situation.

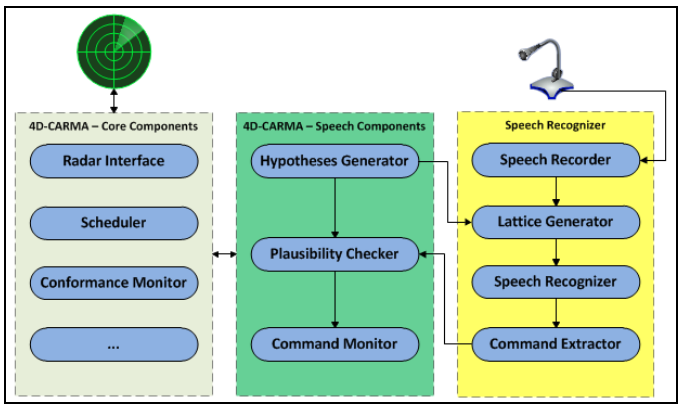

Figure 2 Components of ABSR; in green components of AMAN 4D-CARMA; in yellow components of core speech recognizer (taken from [4])

In [5] command recognition rates (RecR) for ABSR of 95.2% and command recognition error rates(ErrR) below 2% are reported for Dusseldorf Approach Area.

B. *Supervised and unsupervised learning*

Recent advances in machine learning have significantly improved human-machine interaction systems by understanding the context of interaction and adapting to it. Given features describing data and possibly output labels, machine learning aims to model the rules that can map the input features to output labels. Output labels can be categorical, in which the task is called classification, or could be continuous valued in a regression task.

Machine learning methods can be supervised, unsupervised or semi-supervised [20], [21]. In supervised learning, we require data samples and corresponding output labels, and several different algorithms can be used to learn the input output relationship. However, in unsupervised learning, the output labels are missing and the machine learning algorithm just uses the data examples to learn both output labels and the rules to model data. Typical unsupervised learning approaches include data clustering to partition data according an optimization criterion. In semi-supervised learning, partially labeled set of examples are used to build a machine learning model.

Supervised learning approaches utilize different methods to model the input data to output labels. Decision trees [22], [23] use a series of nested rules to compare input data to arrive at the output label. The rules of the tree are pre-determined from expert knowledge or learned from input examples to obtain an optimal partitioning of data using those rules. Decision trees can effectively learn complex input-output relationships with limited data and computational resources. Recently, neural network based models have been shown to accurately learn arbitrary input output relationships. Neural network models require extensive computational resources and are mainly effective when large amount of examples are available, e.g. to build acoustic models for the ABSR system [24].

III. Machine Learning of Command Prediction Models

Assistant Based Speech Recognition (ABSR) normally uses three main models, which need to be trained / adapted for each ATC environment (approach area) separately:

1) Acoustic Model,
2) Language Model (e.g. grammar) and
3) the Command Prediction Model (CPM).

A. Model Interaction within ABSR

Figure 3 shows in the upper part how those three models are used within ABSR. The dark blue ellipses represent the models; rectangles describe tasks and ellipses with a lighter blue show additional inputs and outputs.

At first the CPM is used by the Hypotheses Generator in Figure 3 to derive a set of commands (Command Hypotheses), which are possible in the current situation. These commands are used as input for ASR to reduce the search space size and to guide the search process of the speech recognition system.

The other two models (acoustic and language) are directly used by ASR. For a controller utterance given as audio signal, the acoustic model and the language model are used to extract the sequence of spoken words, e.g. “austrian two zero one descend five thousand feet qnh one zero two two”. A sequence labeling approach is additionally needed to extract the relevant concepts from the recognized word sequence. In this case, one concept is the callsign “AUA201”. The next is the “DESCEND” command with the value “5’000 feet” and we have the concept “QNH” with the value “1022”. These concepts are combined to two recognized commands, here “AUA201 DESCEND 5000 ft” and “AUA201 QNH 1022”. A common ontology for command transcription is being developed by SESAR 2020 exercise PJ 16-04 [25].

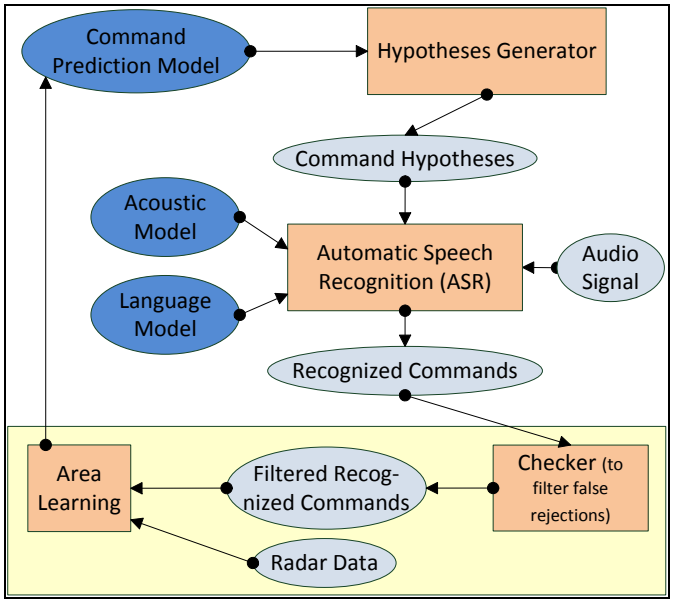

Figure 3 Interaction of Acoustic, Language and Command Prediction Model

From the three models described above, the focus of this paper is on the CPM and how it influences the results of the ABSR.

B. Command Prediction Model as a Decision Tree

Figure 4 shows the structure of the CPM, which is modelled by a decision tree.

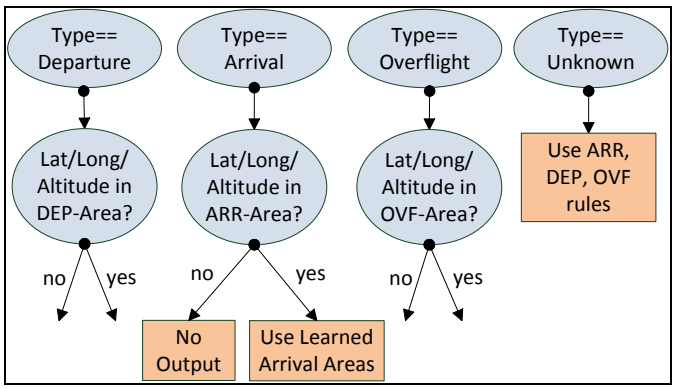

Figure 4 Decision Tree of Hypotheses Prediction Model

For each command type (e.g. DESCEND, REDUCE, INCREASE, CLEARED_ILS) and flight type (e.g. Arrival, Departure, Overflight) a prediction area is needed as shown in Figure 5. If the “Hypotheses Generator” detects that a lat/long position of an aircraft is inside an area of a specific command type, the command values related to that flight and command type are predicted for that aircraft.

Each symbol in the prediction area (see Figure 5) represents a square of 1 nm by 1 nm. These areas can be created manually [26] or learned automatically from transcribed controller utterances and corresponding recorded radar data. This, however, requires either expert knowledge for manual creation and/or expensive manual transcription work of recorded utterances. In order to remove the need of manual work, our approach tries to learn these areas from automatic transcriptions (task "Area Learning" in Figure 3). For each controller utterance the corresponding lat/long positions are known from the recorded radar data, but the (correct) controller commands, however, are unknown. The only things we know are the recognized commands from the Automatic Speech Recognition in Figure 3.

Figure 5 Prediction area of CPM for Cleared ILS-Command for Arrivals

C. *Filtering of Recognized Commands by Checker*

If we have a controller utterance like, "sky_travel two five zero nine descend to flight level nine zero", ABSR should normally recognize the expected command "TVS2509 DESCEND 90 FL". Afterwards this command could be used, together with the corresponding radar data (which amongst others includes flight plan information) for automatic learning of the command prediction model, but the automatic speech recognition could be wrong. That means that the same controller utterance could result in other commands than the expected one, e.g. in "TVS2509 REDUCE 190" or in "DLH109 DESCEND 90 FL". Without further filtering this would either result in an entry of 190 in the area of the REDUCE command or in an entry in the DESCEND area for the correct flight level value, but with the wrong lat/long position, given that DLH109 also has radar data at the same time. To prevent these cases the "Checker" in Figure 3 tries to filter out false recognitions. The challenge for this task is to filter out false recognitions on the one hand, but not to exclude unexpected, but correct transcriptions, on the other hand. To filter out false recognitions, the "Checker" applies the following rules for rejection:

- Commands with unlikely values (e.g. runway that is not available at the airport, values for reduce or descend command to low/high etc.)
- Commands in one transmission that are contrary and usually do not appear together (e.g. turn left and turn right for one aircraft)
- If one of the recognized commands in a transmission is wrong (as described by the rules above) the other commands could be wrong too and will be rejected as well.

D. *Command Prediction Model Learning*

All command recognitions that are not filtered by the "Checker" are included in the Area Learning task in Figure 3. This task does not only mark the areas in which a command type is given, but it also stores the values that occurred for a command type and counts how often a specific command was given in the 1nm by 1nm areas of the CPM (see Figure 5).

If we take a closer look at Figure 5 we can easily see that Cleared ILS commands only occur inside a small area. Two problems become obvious. The model consists only of a limited amount of training data which also contains false recognitions. On the one hand this results in outliers which are probably the result of false recognitions that the "Checker" did not catch. On the other hand there are small gaps between the learned areas where no Cleared ILS command occurred in the training data, but is very likely in reality. To close the gaps and also expand the border of the learned areas we assumed that a valid command that appears at a certain position in the training data is not only valid for this position but also for the surrounding positions. We do not only mark the respective 1*1 nm area in which a command occurs, but also the surrounding areas. An expansion window size of 13 means that we also mark the 168 neighbors (13*13-1) of a lat/long position. In order to reduce the influence of outliers while enlarging the CPM through expansion windows, we added an additional filter starting at a window size of 13x13 nm. The filter removes every 1nm by 1nm area from the CPM that is even after expansion by the window only marked once. The result of this approach for the Cleared ILS area of Figure 5 is shown in Figure 6.

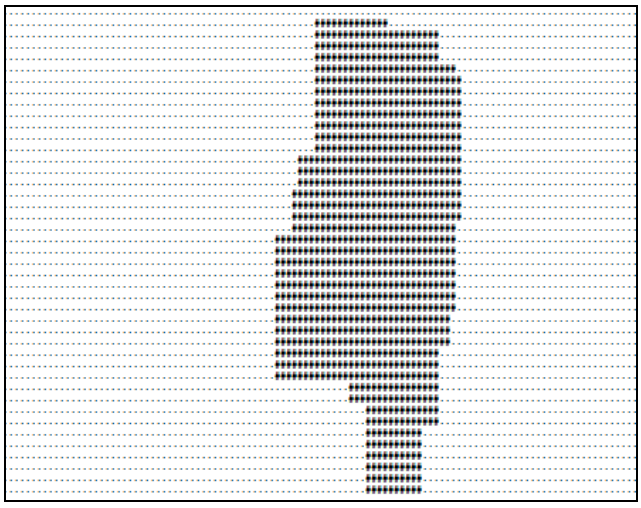

Figure 6 Prediction area of CPM for Cleared ILS-Command for Arrivals (expansion window 13x13 nm)

E. *Iterative Improment of the Recognition Models*

As shown in the bottom yellow shaded part of Figure 3 automatic learning of the predictions areas will result in an im-

proved "Command Prediction Model", which we expect will improve the "Command Hypotheses" iteratively resulting in better "Recognized commands". The aim of the MALORCA project, however, is to learn/improve also the other ABSR models. The "Checker" in Figure 3 helps also to improve "Acoustic Model" and "Language Model", because the learning algorithms for acoustic and language model use the feedback from the additional sensor "Radar data" to decide whether an automatic transcription is good or improvable. In this paper we concentrate on CPM improvement without using inputs from an Arrival Manager as described in Figure 2.

IV. Experimental Set-Up for Proof-of-Concept

The set-up to demonstrate that automatic learning of the CPM is possible and how CPM quality improves with the amount of provided learning data is described now. Radar data for Vienna approach was recorded from July to September 2016 for runway configuration 34 for inbounds and runway configuration 29 and 34 for outbounds. Prague approach data was recorded from August to November 2016 for runway configuration 24 and 30 for inbounds and runway configuration 24 for outbounds. Recordings consist of controller communication to pilots and the corresponding radar data and flight plan information.

We considered four controller positions and learned CPMs for each of them:

1) Prague Arrival Executive Controller (AEC),
2) Prague Director Executive Controller (Feeder, PEC),
3) Vienna sector BALAD executive controller (BALAD),
4) Vienna feeder executive controller (Feeder).

The data was split into two parts. The first part was manually transcribed and used for testing. The majority was automatically transcribed based on the speech recognizer software being developed until May 2017 in the MALORCA project. Currently our training data set for Vienna consists of 18.7 hours of clean speech after removing the silence. For Prague approach 18.1 hours are available.

TABLE 1: COMMAND NUMBER RESULTING FROM AUTOMATIC TRANSCRIPTION

Configuration	# Total Cmds	# Descend cmds	# ILS clearances
AEC	11'103	2'184	351
PEC	5'365	920	458
BALAD	5'929	1'062	13
Feeder	6'959	1'100	245

Table 1 shows the total number of commands resulting from automatic transcription by the acoustic and language model from May 2017 resulting in **RecR** of 89.0% (Prague) and 60.7% (Vienna) without error filtering by the "Checker" in Figure 2. The **ErrR** is 4.1% (Prague) resp. 10.9% (Vienna). **RecR** and **ErrR** will be iteratively improved also by CPM improvement within following MALORCA work packages. Transcribed commands for which automatic transcriptions fails to recognize callsign (output NO_CALLSIGN) or command type (output NO_CONCEPT) are already excluded from Table 1. Correctly transcribed data sets were used to generate the four different command predictions models. We excluded from area learning all *suspicious* recognitions as described in section III.C.

TABLE 2: SIZE OF TEST DATA SET

Approach Area	# Utterances	# given commands	# recogn. commands	# sessions
Prague	2'582	4'563	4'580	27
Vienna	2'427	3'556	3'556	19

For evaluation we used different test utterances which were manually transcribed (see Table 2), i.e. for these utterances the correct transcription (so called gold commands) were known. For each of the 27 resp. 19 controller sessions we calculated different metrics:

- Total number of given commands (**#TgC**),
- Command recognition rate (**RecR**): number of correctly recognized commands divided by #TgC,
- Command rejection rate (**RejR**): number of rejected recognized commands divided by #**TgC**,
- Command recognition error rate (**ErrR**): number of recognized commands which were not spoken and not rejected, divided by #**TgC**,
- **Beta**: number of rejected recognized commands which were included in gold commands, i.e. they were wrongly rejected, divided by number of total rejections,
- Command prediction error rate (**CpER**): number of commands included in gold commands, which were not predicted, divided by #**TgC**,
- Average number of predicted commands (**#NPC**),

We reject a recognized command if it is not predicted by the learned command prediction model. The sum of the values **RecR**, **RejR** and **ErrR** could be greater than 100% (see [4] for detailed definition of the rates), because sometimes more commands are recognized than given. This at least results in an increase of **RejE** or **ErrR**. If more commands are given than recognized, this is always counted as a contribution to **RejR**.

V. Results

In this chapter we present the results of our experiments and compare them to the baseline. As baseline we choose the set of predicted commands when all inputs from automatic data transcription are ignored, i.e. the set of predicted commands includes *all* possible commands.

Table 3 shows the baseline results. It might be confusing that command prediction error rate (**CpER**) is not 0%, if *all* possible commands are predicted. We, however, use some basic heuristics already in this case to reduce number of pre-

dicted commands. The same heuristics are applied when the area based "Command Prediction Model" is used (Figure 5 and Figure 6):

- Altitude commands are only generated in steps of 1000 respectively 10 for flight levels.
- Inbounds get no climb and increase commands, outbounds no descend and reduce commands. Sometimes provided flight plan information is wrong so that this heuristic fails.
- Aircraft do not get commands after a (recognized) handover command.
- Predicted QNH and ATIS INFORMATION command values (e.g. 1013 or charly) depend on values used in previous commands.
- Only IFR flights which are in a defined polygon around the airport are considered (Vienna test radar data on average contains 70 aircraft whereas on average only for 29 aircraft commands are predicted.)

If these hypotheses are not valid a command prediction error is observed. These errors and others are observed as well when area based CPM is used (Figure 5). The only exception is that altitude values which are no multiple of 1000 (e.g. 2700 or 3400 feet) are learned from automatic transcription. To conclude: Area based command prediction will at least include 3.7% (Vienna) resp. 1.1% (Prague) **CpER**.

TABLE 3: METRICS IF ALL COMMANDS ARE PREDICTED (BASELINE)

	#TgC	**RecR [%]**	**ErrR [%]**	**RejR [%]**	**Beta [%]**	**CpER [%]**	**#NPC**
Vienna	3556	60.4	3.7	36.1	0.8	3.6	11505
Prague	4566	88.1	1.1	11.0	10.0	1.0	2054

In a first approach for automatic learning of the CPM we just used the automatically recognized commands (filtered commands are excluded) and stored the areas in which those commands occurred, according to the aircraft radar data (see chapter III). Table 4 shows the evaluation results of this approach.

TABLE 4: METRICS FOR SIMPLE COMMAND PREDICTION MODEL

	#TgC	**RecR [%]**	**ErrR [%]**	**RejR [%]**	**Beta [%]**	**CpER [%]**	**#NPC**
Vienna	3556	39.4	1.4	59.4	34.5	35.2	924
Prague	4566	70.3	0.4	29.6	63.3	21.5	350

Compared to our baseline the average number of predicted commands is reduced by a factor of 12 resp. 6, but this reduction comes with a price. The reduced set of predicted commands drops the **RecR** for the Vienna test data to 39.4% (baseline 60.4%) resp. 70.3% (baseline 88.1%) for Prague.

The poor quality of the CPM and the resulting large loss in **RecR** comes from the limited amount of training data. To improve the quality of the CPM we used the window based approach (see section III.D). We experimented with different expansion windows from 3x3 nm to 29x29 nm. The results of this approach are shown in Table 5 (Vienna) and Table 6 (Prague).

TABLE 5: VIENNA - METRICS FOR WINDOW BASED CPM

	#TgC	**RecR [%]**	**ErrR [%]**	**RejR [%]**	**Beta [%]**	**CpER [%]**	**#NPC**
3x3	3556	51.9	1.68	46.6	17.6	16.60	1026
5x5	3556	54.8	1.75	43.6	12.4	12.30	1104
7x7	3556	56.8	1.90	41.5	9.0	9.61	1170
9x9	3556	57.9	1.93	40.4	6.7	8.22	1229
11x11	3556	58.4	1.96	39.8	5.6	7.40	1274
13x13	3556	58.1	1.93	40.1	6.1	8.14	1215
15x15	3556	58.5	1.93	39.8	5.4	7.59	1246
17x17	3556	58.6	1.96	39.6	5.0	7.24	1277
19x19	3556	58.9	1.98	39.3	4.4	6.89	1304
21x21	3556	59.0	1.98	39.1	4.1	6.74	1328
25x25	3556	59.1	2.01	39.1	4.0	6.44	1368
29x29	3556	59.2	2.22	38.7	3.8	6.21	1400

By enlarging the CPM through the window based approach we observe an increase of **RecR**, but also an increase of **ErrR** and average number of predicted commands (**#NPC**). If we put the gain in **RecR** against the loss in **ErrR** we can see a benefit up to a window size of 25x25 for Vienna and Prague. At this window size we still have a gain in **RecR** of 0.1% for Vienna and Prague, but only a loss in **ErrR** of 0.03% for Vienna resp. 0.00% for Prague.

TABLE 6: PRAGUE - METRICS FOR WINDOW BASED CPM

	#TgC	**RecR [%]**	**ErrR [%]**	**RejR [%]**	**Beta [%]**	**CpER [%]**	**#NPC**
3x3	4566	82.7	0.68	16.9	36.9	8.02	464
5x5	4566	85.3	0.70	14.3	25.2	5.13	511
7x7	4566	86.0	0.70	13.5	21.1	4.19	541
9x9	4566	86.6	0.77	12.9	18.2	3.57	562
11x11	4566	86.9	0.77	12.6	16.6	3.21	576
13x13	4566	86.5	0.76	13.0	18.8	3.78	552
15x15	4566	86.8	0.80	12.7	16.8	3.48	564
17x17	4566	87.1	0.80	12.3	15.5	3.05	574
19x19	4566	87.4	0.83	12.1	14.6	2.80	583
21x21	4566	87.6	0.85	11.9	13.5	2.56	591
25x25	4566	87.7	0.85	11.7	12.5	2.35	604
29x29	4566	87.8	0.93	11.6	12.4	2.29	615

If we compare the results of the 25x25 window to our baseline the loss in **RecR** is 1.3% (Vienna) resp. 0.4% (Prague) and the improvement in **ErrR** is 1.69% (Vienna) resp. 0.25% (Prague). The difference between baseline and the window based approach is relatively small. The benefit of the window based approach becomes visible, when we take the predicted command set size into account. Compared to the baseline the window based approach delivers fewer predicted commands. For Vienna and Prague we get an average #**NPC** of 1368 resp. 604. That means a reduction of the predicted command set size

by a factor of 8 resp. 3 compared to baseline with all possible commands predicted. The smaller the predicted command set size is, the better the output of the speech recognizer and the better the filtering of the "Checker" will be. The search lattice size (see explanation of Figure 2) exponentially depends on predicted command set size.

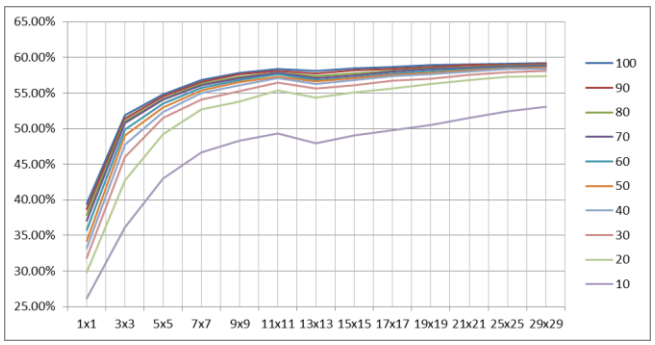

Figure 7 Dependency of ReCR from training data size (Vienna)

The acceptable size of the predicted command set that is applicable depends on the application and how fast it has to update. For an application in which the predicted command set only significantly changes every 60 seconds, a bigger predicted command set is not an issue.

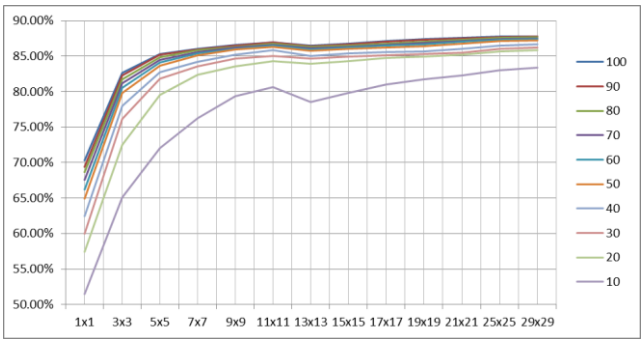

Figure 8 Dependency of RecR from training data size (Prague)

To determine the relevance of training data for a CPM we trained the model with different amounts of data from 10% to 100% of the available training data. We executed those evaluations for all window sizes for Vienna and Prague. The results for **RecR** can be seen in Figure 7 and Figure 8. If we look at the results with only 10%, 20% and 30% of the training data, there is a relatively big increase in recognition rate especially with small window sizes. This difference decreases with larger window sizes, because the larger windows compensate some of the missing data. With more training data the increase in **RecR** gets visibly smaller, but if we look at the difference between 90% and 100% of the data, we still get an increase in **RecR** of 0.14% (Vienna) resp. 0,02% (Prague) with a window size of 29x29. That does not seem like a large improvement, but we have to take into account that the 29x29 window already compensates for a huge amount of not available training data. To conclude: With more training data a small improve in **RecR** is still possible. Also the window size for the CPM could be reduced with more training data, since the need to compensate missing training data would be smaller.

VI. Discussion of Results And Next Steps

The output of the current version of the acoustic and language model are quite noisy with respect to command recognition (**RecR**) and command error rate (**ErrR**). Although we have only RecR of 60.7% resp. 89.0% in the learning data, the learned models for the checker could reduce ErrR by a factor of 5 for both Vienna and Prague approach (from 10.9% to 2.0% for Vienna resp. 4.1% to 0.8% for Prague). Obviously the size of training data also increases the **RecR**, which is of course limited by the recognition rate without using the "Checker".

Figure 7 and Figure 8 shows a logarithmic dependency of recognition rate from data size (ds). If we assume the relation

$$ds = m*\ln(RecR) + b$$

we could expect an increase of **RecR** for Vienna from 58.6% to 59.4% by increasing recorded speech data size by a factor of 2 (i.e. from 18.7 to 35.4 hours) and of 61.6% if we increase by a factor of 10 (window size 17x17 nm). For Prague we could expect an increase from 87.1% to 88.3% (two times more data) resp. 90.9% (10 times more data). These numbers should just show that more data can improve recognition quality, but the effects will be even smaller, because **RecR** after rejection could not be better than the **RecR** without rejection.

If data size is limited (which is always the case) we can also improve recognition rate by increasing the window size which emulates the availability of more data. Increasing window size also increases the error rate, because the areas of outliers and false rejection are also enlarged (see Figure 6). Currently we are improving our window model by (1) introducing a dynamic filter for outliers (not always just remove fields which are marked once), (2) varying the effect the expansion window has on surrounding areas (not increase all areas in the window by 1.0) and (3) using expansion windows which are command type dependent. The area for an ILS clearance is much smaller than for a descend clearance. Therefore, smaller windows are more suitable for the ILS clearance. On the other hand the area for a reduce command is comparable to the area size of a descend command. We, however, only have 450 commands for learning the reduction area, but 2'150 command for learning the descend area. Therefore, window size should also depend on number of available training data for a command type.

Recognition rates of 60% (Vienna) resp. 88% (Prague) seem to be low compared to 95% reported by the AcListant® project [5], [6]. We should, however, keep in mind, that we only concentrated on the "Checker" component. An improved CPM also improves the ABSR output itself (yellow part in Figure 2) and it will help to improve acoustic and language model improvement which will result in better inputs for CPM learning.

VII. CONCLUSIONS

We presented an algorithm which automatically learns a model to predict radar approach controller commands using only radar data, flight plan information and recorded untranscribed controller utterances. Compared to a neural network based approach resulting in a black box model the presented model is based on a decision tree. The command prediction model (CPM) was validated against transcribed controller commands for Vienna and Prague approach for the feeder and sector position, i.e. four command prediction models were learned. In this study the CPM was used for filtering the output of an automatic speech recognizer with low performance (89.0% for Prague and 60.7% for Vienna), i.e. for rejecting wrongly recognized commands. The presented machine learning based algorithm for controller command prediction was successfully validated: Command error rates could be reduced from 4.1% to 0.9% for Prague approach and from 10.9% to 2.0% for Vienna approach.

Compared to AcListant® project with recognition rates of 95% the presented recognition rates seem quite low. This study, however, only uses pure radar and flight plan data and does not use the outputs of an arrival manager, which also contributed to the high recognition rate in the AcListant® project. Furthermore the improved CPM will improve machine learning of acoustic and languages models again resulting in an improved CPM.

Overall, the impact of the solutions of the MALORCA project when integrated into the current ATM procedures is expected to be high, especially due to minimizing the total costs related to the implementation of decision and negotiation support systems and related to the maintenance and system changes towards new ATM procedures.

ACKNOWLEDGMENT

We would like to thank all the controllers who anonymously provided us with real world command examples and also our MALORCA partners from Austro Control and from Air Navigation Service Provider of Czech Republic.

REFERENCES

[1] MAESTRO AMAN from Thales Groupe https://www.thalesgroup.com/en/worldwide/aerospace/topsky-atcmaestro, n.d.

[2] The project AcListant® (Active Listening Assistant) http://www.aclistant.de/wp, n.d.

[3] H. Helmke, H. Ehr, M. Kleinert, F. Faubel, and D. Klakow, "Increased Acceptance of Controller Assistance by Automatic Speech Recognition," in 10th USA/Europe Air Traffic Management Research and Development Seminar (ATM2013), Chicago, IL, USA, 2013.

[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", in 11th USA/Europe Air Traffic Management Research and Development Seminar (ATM2015), Lisbon, Portugal, 2015.

[5] H. Helmke, O. Ohneiser, Th. Mühlhausen, M. Wies,."Reducing Controller Workload with Automatic Speech Recognition", in IEEE/AIAA 35th Digital Avionics Systems Conference (DASC). Sacramento, California, 2016.

[6] H. Helmke, O. Ohneiser, J. Buxbaum, Chr. Kern, "Increasing ATM Efficiency with Assistant-Based Speech Recognition", in 12th USA/Europe Air Traffic Management Research and Development Seminar (ATM2017). Seattle, Washington, 2017.

[7] H. Helmke, "Grant Preparation: MALORCA – Part B, version 0.16, 18 March 2016 and AcListant project: 'Feedback of Austro Control controllers during debriefing sessions'," Braunschweig, December 2014.

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

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

[10] Y. Oualil, D. Klakow, G. Szaszák, A. Srinivasamurthy, H. Helmke, P. Motlicek, "Context-aware speech recognition and understanding system for air traffic control domain", to be published in IEEE Automatic Speech Recognition and Understanding Workshop (ASRU 2017), Okinawa, Japan, Dec. 2017, to be published.

[11] C. Hamel, D. Kotick, and M. Layton, "Microcomputer System Integration for Air Control Training," Special Report SR89-01, Naval Training Systems Center, Orlando, FL, USA, 1989.

[12] J.M. Cordero, N. Rodríguez, J.M. de Pablo, and M. Dorado, "Automated Speech Recognition in Controller Communications applied to Workload Measurement," 3rd SESAR Innovation Days, Stockholm, Sweden, 2013.

[13] S. Chen, H.D. Kopald, A. Elessawy, Z. Levonian, and R.M. Tarakan, "Speech Inputs to Surface Safety Logic Systems," IEEE/AIAA 34th Digital Avionics Systems Conference (DASC), Prague, Czech Republic, 2015.

[14] S. Chen, H.D. Kopald, R. Chong, Y. Wei, and Z. Levonian, "Read Back Error Detection using Automatic Speech Recognition," 12th USA/Europe Air Traffic Management Research and Development Seminar (ATM2017), Seattle, WA, USA, 2017.

[15] S.R. Young, W.H. Ward, and A.G. Hauptmann, "Layering predictions: Flexible use of dialog expectation in speech recognition," in Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI89), Morgan Kaufmann, 1989, pp. 1543-1549.

[16] S.R. Young, A.G. Hauptmann, W.H. Ward, E.T. Smith, and P. Werner, "High level knowledge sources in usable speech recognition systems," in Commun. ACM, vol. 32, no. 2, Feb. 1989, pp. 183-194.

[17] D. Schäfer, "Context-sensitive speech recognition in the air traffic control simulation," Eurocontrol EEC Note No. 02/2001 and PhD Thesis of the University of Armed Forces, Munich, 2001.

[18] A. Schmidt, "Integrating Situational Context Information into an Online ASR System for Air Traffic Control," Master Thesis, Saarland University (UdS), 2014.

[19] 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.

[20] T. Hastie, R. Tibshirani, J. Friedman, "The Elements of Statistical Learning: Data mining, Inference, and Prediction," New York: Springer, 2009, pp. 485–586.

[21] R. O. Duda, P. E. Hart, D. G. Stork, "Unsupervised Learning and Clustering," Pattern classification (2nd ed.), Wiley, 2001.

[22] J. R. Quinlan, "Induction of Decision Trees, Machine Learning," Vol 1(1), pp. 81-106, Kluwer Academic Publishers, 1986.

[23] I. H. Witten, E. Frank; M. A. Hall, "Data Mining: Practical Machine Learning Tools and Techniques," (3rd ed.), Elsevier, 2011.

[24] G. Hinton, L. Deng, D. Yu, G. Dah et al., "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups, " IEEE Signal Processing Magazine, Vol. 29, pp. 82-97, 2012.

[25] H. Helmke, M. Slotty, M. Poiger, D. F. Herrer, O. Ohneiser et al., "Ontology for Transcription of ATC Speech Commands," 7th SESAR Innovation Days, Belgrade, 2017, to be published.

[26] M. Hössl, H. Helmke, J. Gottstein, "Why Controllers Seldom Stick to the Book and How Their Commands are Predictable Nevertheless," in ICRAT conference, Istanbul, May. 2014.