Automated Interpretation of Air Traffic Control Communication: The Journey from Spoken Words to a Deeper Understanding of the Meaning

Matthias Kleinert[1], Hartmut Helmke[1], Shruthi Shetty[1], Oliver Ohneiser[1], Heiko Ehr[1],
Amrutha Prasad[2], Petr Motlicek[2], Julia Harfmann[3]

[1]German Aerospace Center (DLR), Institute of Flight Guidance, Braunschweig, Germany
[2]Idiap Research Institute, Martigny, Switzerland,
[3]NATS (Enroute) PLC, Whitely, Fareham, Hampshire, United Kingdom

matthias.kleinert@dlr.de, hartmut.helmke@dlr.de, shruthi.shetty@dlr.de, oliver.ohneiser@dlr.de,
heiko.ehr@dlr.de, amrutha.prasad@idiap.ch, petr.motlicek@idiap.ch, julia.harfmann@nats.co.uk

2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC) | 978-1-6654-3420-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/DASC52595.2021.9594387

***Abstract*—Sophisticated Automatic Speech Recognition (ASR) technologies have become increasingly popular and are widely used in all domains over the years. Systems like Google Assistant, Siri®, Alexa® are integrated into our day-to-day lives. These systems offer a wide range of possible applications just by understanding human speech. However, in the Air Traffic Control (ATC) domain, even the most advanced simulators can just partially replace expensive pseudo-pilots. In spite of having a standardized ATC phraseology, it is still a major challenge to recognize and correctly understand the communication between air traffic controllers (ATCo) and pilots. This is because understanding an ATCo-pilot communication requires more than just transforming speech to a sequence of words. For most ATC applications, perfectly recognizing the sequence of words would not be useful, if the meaning behind the word sequence cannot be correctly interpreted. Recently, 20 European partners from Air Traffic Management (ATM) domain have agreed on a common set of rules, i.e., an ontology on how to transform the spoken words into ATC instructions that clearly define the meaning of the words and make them usable for different applications. In this paper, we present an extension of the mentioned ontology to make it usable for pilot speech as well. We also show some of the challenges faced in understanding the meaning of ATCo-pilot communication and describe our approach of tackling them. Furthermore, we present an algorithm to transform words automatically into ontology instructions and describe the interfaces used to ensure a consistent and reliable communication of ATC instructions. This interface includes, besides other information, plausibility values, different speakers, and ambiguous outputs.**

Keywords—air traffic control, ATC command ontology, command extraction, language understanding, command recognition rate, JSON

I. INTRODUCTION

Automatic Speech Recognition (ASR) and other related technologies are being used in a broad variety of applications. The applications range from recognizing human speech for documentation purposes to more complex ones that also require some understanding of human speech, e.g., control of home automation via voice with systems such as Google Assistant or Alexa®. In the field of Air Traffic Control (ATC), recognition and understanding of communication between air traffic controller (ATCo) and pilot has not been addressed in a sophisticated manner yet. The only application of ASR in this area being already implemented is a partial replacement of pseudo-pilots through speech understanding in some simulation exercises [1]. Even with the standardized ATC phraseology, it is still considered a major challenge to recognize and understand the communication between ATCo and pilot as understanding requires much more than just transforming a spoken utterance into a sequence of words.

Natural human language offers a lot of possibilities to express a certain intention with many different words and ATC communication is no exception to this. Even though ATC phraseology is standardized, ATCos express the same command in many different ways using different vocabulary. This is illustrated in the following list:

- lufthansa three echo romeo make it heading two two zero degrees left
- echo romeo turn left two two zero degrees
- lufthansa three echo romeo turn left now onto a heading of two two zero
- three echo romeo turn left now heading two two zero degrees
- lufthansa three echo romeo heading left of two twenty
- lufthansa three echo romeo turn further left heading two two zero degrees

The wording of all sentences is different, but the meaning is always the same. The aircraft with the callsign DLH3ER should change its flight direction to a heading of 220 degrees. A human with a certain amount of ATC knowledge would be able to identify that all the above-mentioned sentences mean the same. For a machine this is a complex task and it gets more complex when multiple commands and lots of different speakers are involved. This means, that for most ATC applications the recognition of the spoken word alone would not be useful, if the meaning behind it is unknown. A transformation of spoken words into meaningful concepts is required, which maps spoken words to standardized concepts. This was already achieved by 20 European partners from the Air Traffic Management (ATM) domain, who defined a common set of rules (ontology). This ontology (1) defined the important conceptual elements of ATC voice transmission and (2) allowed a clear interpretation of the meaning amongst different applications

[2]. As per this ontology all the above listed sentences would result in the abstract concepts "DLH3ER HEADING 220 LEFT". *DLH3ER* is the callsign of the addressed aircraft. *HEADING* is the command type, *220* is the value of the command, and *LEFT* is the qualifier. One can imagine that for a machine this is a format which can be interpreted more easily.

ATCo transmissions were the main focus of the initially presented ontology, but the current SESAR project HAAWAII [3] recently extended the original ontology to make it also applicable for voice utterances of pilots. This does not only enable an interpretation of the meaning of pilot utterances, it furthermore allows the comparison of ATCo and pilot transmissions on a conceptual level. Applications such as read back error detection, which are almost impossible to accomplish on word level, now seem possible. The extension of the ontology is presented in section III.

The definition of the ontology is important, but it does not provide its full benefit until it can be applied and transformed automatically into a format that can be exchanged among different systems and applications. Therefore, this paper also focuses on a technical implementation of the ontology. This implementation first covers the automatic transformation of recognized words from ATCo/pilot voice utterances into ontology-based concepts. The second part of the implementation describes the used machine readable JSON (=JavaScript Object Notation) ontology format, which allows the transmission of information among all kind of applications and is easy to expand.

In the next section, we present related work. Section III shows the extension of the original ontology for ATC voice communication. In section IV the algorithm to transform the spoken words into ontology concepts is shown in more detail. The machine readable JSON format for the ontology is presented with examples in section V. Section VI and section VII show the experimental setup and results, which proves the capability of the automatic ontology transformation. We conclude the paper in section VIII and give an outlook on future work.

II. Related Work

The ontology for ATC instructions, first introduced by the CWP HMI project [2] is not final yet, which means that updates/changes are still expected. The projects STARFiSH [4] and "HMI Interaction Modes for Airport Tower" [5] expand the ontology with respect to ATC ground and tower commands including remote tower operations. The projects "HMI Interaction modes for approach control" [6] and HAAWAII [3] also include pilot utterances as well as enroute and oceanic traffic. Furthermore, HAAWAII uses ASR to predict ATCo workload. Therefore, greetings – it is yet to be investigated whether they are more likely to be omitted or elongated in high workload situations – are important and introduced to the ontology as well.

One of the first publicly available corpora with transcribed speech recordings for the ATC domain was the ATCOSIM corpus, which was funded by Eurocontrol [7]. Our transcription rules for writing down the utterances word by word are very similar, but in addition to [7] we propose also rules for the annotation. Nguyen and Holone [8], [9] proposed 10 classes to replace word sequences with their corresponding class label, e.g., callsign, unit-name, fix, number. Johnson et al. [10] proposed a keyword and value representation in JSON format [11], where keywords could be Callsign, ToFix, FlightLevel, Altimeter, etc.

In the AcListanct® project [12], Saarland University and DLR created an ontology which consists of the four elements callsign, command type, command value, and unit [13], [14]. Similar to ATCOSIM the ATCO^2 project aims to develop a unique platform allowing to collect, organize, and pre-process air traffic control (voice communication) data from different airspaces [15].

The HAAWAII project addresses readback error detection [3]. The communication feedback loop defines that ATCos transmit verbal ATC instructions via radiotelephony, whose safety-related parts need to be read back by pilots. ATCos need to hear back pilot readbacks and correct the readback in case of errors [16]. A hearback error is a readback error which is undetected by the ATCo and is left uncorrected.

Fortunately, communication errors, which include readback and hearback errors occur very seldomly in ATC. Depending on the definition of an error and the analyzed airspace, the occurrence of errors in ATC communication varies between less than 1% and up to 7% [17], [18], [19], [20]. Some transmissions even containing multiple errors [21]. NASA aviation safety reporting system reports blame communication errors being at least contributing to 80% of incidents or accidents [22]. EUROCONTROL assumes that miscommunication is the reason for roughly 30% of incidents [23]. These numbers and references clearly show that the content of communication between ATCos and pilots is of utmost importance for the safety of air traffic resulting in the importance of automatic understanding of ATCo pilot communication, i.e., not just recognizing the spoken words, but really addressing the semantic level.

III. Meaning Of The Spoken Words

This section presents the ontology originally defined in the SESAR PJ.16-04 CWP HMI project together with the extension for pilot speech from the current SESAR HAAWAII project. Fig. 1 presents the general structure of the different elements in the ontology. The highest conceptual element in this definition is an instruction and according to the ontology rules a voice utterance can consist of either one or multiple instructions. Fig. 1 depicts the structure of an instruction and shows that an instruction consists of a callsign, a command, and optional conditions. A command always has a type, which determines, how many values are allowed. Optional fields are unit (e.g., FL, ft, kt), qualifier (e.g., LESS, OR_BELOW, LEFT), speaker (PILOT or empty), and reason (REQUEST, REPORTING or empty).

The first element of an instruction is always the *callsign* (or *NO_CALLSIGN*), independent of where in the speech utterance it is pronounced. Even though the callsign is usually said only once in a voice utterance, on the conceptual ontology level it is specified to be repeated in front of every instruction in case the speaker uses multiple commands. The ontology also defines how to use contextual information on active aircraft in the working area. So, if in an utterance a callsign is shortened by leaving out some numbers, letters or the airline designator, the complete callsign is always represented on the instruction level, if possible. For example,

if only "air france three delta" is said or recognized in the context where no AFR3D exists, but AFR123D exists, the callsign should be automatically corrected to AFR123D on ontology level. This compensates for misrecognitions on word level and deals with commonly used abbreviations for callsigns in ATC. In case a callsign cannot be determined or was not said at all, the special keyword "*NO_CALLSIGN*" is used.

The other ontology elements from Fig. 1 are now explained in more detail via real world examples from different airspaces and airports. An example from approach traffic could be:

speed bird six nine six victor keep speed one six zero knots until four miles final

As per ontology, this would result in "BAW696V MAINTAIN SPEED 160 kt UNTIL 4 NM FINAL". The *type* of this command is "MAINTAIN SPEED", so a type can consist of one or two words, but not more. The *value* is "160" and the *unit* in this case is "kt". The last four elements after "kt" are the *conditional clearance* with the conjunction "UNTIL" and the requirement "4 NM FINAL".

A ground-based example from the tower area could be:

lufthansa four nine nine taxi to alfa five eight via lima and november eight

This would result in two instructions "DLH499 TAXI TO STAND_A58" and "DLH499 TAXI VIA L N8". As described before the callsign is present in every instruction even though it was said only once. Furthermore, the command type "TAXI TO" can have by definition only one value ("STAND_A58"). The definition of the "TAXI VIA" command also allows multiple values, in this case "L" and "N8". Here, the knowledge of "alfa five eight" being a stand and "lima november eight" being taxiways should be known beforehand. For this purpose, the ontology defines a configuration file, which specifies special word sequences and their mapping to values in the ontology. With this solution it is possible that *"alfa five eight"* is mapped to "STAND_A58" even though it is not known from the recognized words that this word sequence belongs to a parking stand.

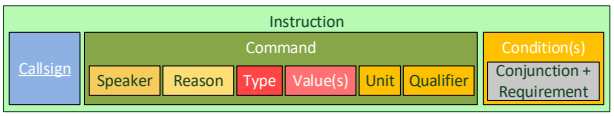

Fig. 1 Elements of an instruction consisting of a callsign, a command, and condition(s).

To demonstrate the ontology extension of HAAWAII we make it a bit more complex and look at a communication between ATCo and pilot. The following example is taken from enroute traffic:

Pilot: reykjavik control [NE Icelandic] godan dag [/NE] iceair six eight lima passing level one nine zero climbing two nine zero

ATCo: [unk] six eight lima reykjavik control [NE Icelandic] godan dag [/NE] identified climb to flight level three seven zero

The communication starts with the pilot and the transcription/recognition contains some special markings such as "[NE]" (meaning "Non-English") or "[unk]" (meaning "unknown"). The updated ontology specifies that text parts of the transcription which are identified as non-English have to be enclosed within "[NE] [/NE]" tags. Furthermore, if it is possible to identify the language spoken, this information also needs to be mentioned, e.g., *"Icelandic"*. The above utterance results in the following ontology instructions:

```
ICE68L PILOT   STATION REYK_RADAR
ICE68L PILOT   GREETING
ICE68L PILOT   REPORTING ALTITUDE 190 FL
ICE68L PILOT   REPORTING CLIMB 290 none

ICE68L         STATION REYK_RADAR
ICE68L         GREETING
ICE68L         INIT_RESPONSE
ICE68L         CLIMB 370 FL
```

The new *speaker* field is only used, if the speaker is not the ATCo to ensure compatibility with the original ontology version. Also shown here, in cases where a report or a clearance does not contain a unit, the ontology captures this information with a "none" in the unit field. This is visible in the given example for the "CLIMB" command which is once transformed with "FL" and once with "none". The *reason* field also belongs to the HAAWAII extension of the ontology and is only used for pilots. "REPORTING" as shown in the example indicates that something is being reported instead of being requested. This information for example becomes important for the decision if a readback is required or not. The decision regarding a command(s) being a report, a request or a readback is not always easy. The utterance *"descending flight level two five zero"* from a pilot for example can be an altitude readback or a report. Both "ICE68L PILOT REPORTING DESCEND 250 FL" and "ICE68L PILOT DESCEND 250 FL" are, therefore, possible. One could easily determine which one is correct by looking into the previous utterances. The ontology definition, however, requires considering only the current utterance for creating the instructions. Not all words from recognitions are mapped to elements from the ontology. For example, the utterance *"okay we check thanks air canada eight five four"* results in "ACA854 NO_CONCEPT". There can be two reasons for this, either the spoken words are not important in the context of ATC communication or there is no fitting element yet to cover the words. The second option is a hint for possible extensions of the ontology. However, NO_CONCEPT is only presented in the ontology format if no other command can be extracted from the words. The utterance *"okay we check thanks air canada eight five four descend three thousand feet"* would, therefore result in "ACA854 DESCEND 3000 ft". An implementation of the ontology from DLR already exists, which includes an automatic extraction (command recognition) from word sequences to the ontology concepts. The next section describes how this extraction is done.

IV. From Words To Meaning With Ontology

The basic *Command Extraction* algorithm was already presented in 2020 [24]. It tries to identify patterns occurring in ATCo and pilot utterances. It 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. 2 illustrates the complete command extraction algorithm:

```
01  Extract Predicted Callsign from transcription;
02  while (NOT end of utterance reached)
03 {
04    Extract command when matching keyword sequence found;
05    if (value, unit, qualifier ... could be extracted)
06    {
07      Add command to extracted commands;
08    }
09 }
10 while (NOT end of utterance reached)
11 {
12   Find ATC concept in unclassified words;
13   if (unit, qualifier ... could be extracted)
14   {
15     Add command to extracted commands;
16   }
17   Find command hints in unclassified words;
18   if (value, unit, qualifier ... could be extracted)
19   {
20     Add command to extracted commands;
21   }
22 }
23 while (NOT end of utterance reached)
24 {
25   Find first or further callsigns in unmatched words;
26 }
27 Find commands from unclassified numbers
```

Fig. 2 Callsign and Command Extraction Algorithm.

The algorithm has already been explained with examples in [24]. The last line is newly added, which tries to find commands by extracting commands from sheer numbers, which are currently not mapped to any command.

This could be best explained by using an example of a pilot utterance: "*one hundred left heading one two zero speed bird five zero two papa".* First the callsign "*BAW502P*" is extracted in line 1. Lines 4-9 then extract "*PILOT HEADING 120 LEFT*". After this, nothing is further extracted until line 26.

TABLE I shows the status of command extraction, when line 27 of the algorithm is reached.

TABLE I. ANNOTATIONS FOR ONE SPEAKER

one	hundred	left	heading	one	two	zero	speed	bird	five	zero	two	papa
unkn	unkn	qual	type	valu	valu	valu	csgn	csgn	csgn	csgn	csgn	csgn

Here, we see that the words "one" and "hundred" are classified as "unkn" (unknown), which means that they are still unrecognized. This could be a speed, a heading, an altitude in feet or a flight level, a frequency, a mach number, a QNH value or a squawk code. However, out of these valid candidates for a value of 100 are speed, flight level and heading, because 100 does not fit all the other types. Altitude values in feet are usually between 1,500 and 10,000, and QNH values between 950 and 1080 etc. Depending on the airport and the aircraft type and its position, a speed of 100 knots can also be excluded. Speed advisories are usually above 150 knots, even in final approach. Nevertheless, "one hundred" could mean a flight level or a heading. In either case, "one zero zero" is preferred by the ICAO phraseology. We use the heuristic to compare with the extracted commands from the previous utterance, in order to determine the command type for a value of 100.

Extracting commands from pilot utterances as in the example is not an easy task. The reason for this is that pilots most often deviate from the ICAO phraseology and tend to use short forms while communicating with the ATCo. However, we can benefit from the knowledge, that the example utterance is a pilot readback, which means that there exists a previous utterance from the ATCo for this callsign. The command extraction model should also correctly recognize the value 100 and determine if it is a HEADING or an ALTITUDE command or both. If both command types with a value of 100 are returned, then the extraction is aborted, because it would be ambiguous. Otherwise, we extract either a HEADING or an ALTITUDE command with the value of 100. It should also be noted that for an altitude value, the ATCo command could also be a CLIMB, DESCEND, STOP_DESCEND or STOP_CLIMB command. In all cases, pilots could just read back "one hundred", which is not recommended by the ICAO phraseology.

In the above example, this ambiguity regarding which command the value belongs to can be resolved only if the extracted commands for the corresponding ATCo utterance contains it as shown in the example ATCo annotation below.

- *"BAW502P HEADING 120 LEFT*" and
- *"BAW502P DESCEND 100 FL*".

Section VII describes the results when line 27 of the algorithm is used and when not used.

V. TECHINCAL ONTOLOGY IMPLEMENTATION

The SESAR solution PJ.16-04-ASR has defined an ontology for transcription and annotation of ATCo-pilot communication, which is extended by several European ASR projects defined in the related work section. The resulting sequences of spoken words and the corresponding ATC concepts are easily readable by human experts, which eased the agreement on a common ontology within PJ.16-04 solution. It is, however, much more difficult for a computer to extract and interpret consistently from regular text sequences, what is a callsign, a command type, a qualifier etc. Therefore, the next logical step is to extend the ontology with a format to ease automatic interpretation. This section presents a machine-readable format based on JSON, which is used by HAAWAII, STARFiSH, PJ.10-97-ASR and PJ.10-96-ASR projects. The JSON format consists of key-value pairs. The keys are always enclosed in quotations marks, whereas the values could be strings, numbers or the Boolean values *true* and *false*. The used JSON formats are explained for different use cases by examples in the following subsections.

A. JSON format for the transcriptions

The example in TABLE II shows the output of Speech-to-Text transformation, i.e., the transcription or recognition, in JSON format. It contains at least the name of the wave file with the corresponding voice recording and the sequence of words of the spoken utterance in the agreed format.

TABLE II. SPEECH-TO-TEXT OUTPUT – MINIMAL VERSION

```
{
   "filename": "2020-11-20__13-03-13-24.wav",
   "speaker" : "ATCo",
   "abstraction_layer_word_sequence":
        "oscar tango whiskey you may leave [NE Austrian] servus [/NE]"
}
```

This format is used to encode both the output of manual transcriptions and the output of automatic speech-to-text transformation. Additional keys include *systemtimetick*, *speaker* (see table above), *sender, filename*. We subsume the example from TABLE II by the definition in TABLE III. The content of FILE_DESC is shown by example in TABLE II and can also contain the additional keys mentioned above. The non-terminal UTTERANCE is detailed more in the next example in TABLE IV.

TABLE III. SPEECH-TO-TEXT OUTPUT – FORMAL DEFINTION

```
{
   FILE_DESC
   "abstraction_layer_word_sequence": UTTERANCE
}
```

The first example in TABLE II contains a simple utterance where only one speaker was recorded. But speaker recordings could also contain utterances belonging to multiple speakers consisting for example of a pilot reporting, an ATCo responding and another pilot calling in like shown in TABLE IV. The description below the table shows the definition of the format in form of a context free grammar.

TABLE IV. UTTERANCE FROM ATCO AND PILOT

```
{
"abstraction_layer_word_sequence":  [
    { "recog_speaker": "Pilot1", "recog_utter":
        "tarom three four one x-ray fully established at seven miles" },
    { "recog_speaker": "ATCo", "recog_utter":
                    "tarom three four one x-ray …"},
    { "recog_speaker": "Pilot2", "recog_utter":   "good morning" }
]
}
```

UTTERANCE = WORD_SEQ | WORD_SEQ_ARR
WORD_SEQ_ARR = [SPK_WORDS, SPK_WORDS* …]
SPK_WORDS = {"recog_speaker": SPK, "recog_utter": WORD_SEQ}
SPK = "ATCo" | "Pilot" | "Pilot1", "Pilot2" …

In the above description, "|" means "or", the asterix symbol "*" means zero or more times, i.e., WORD_SEQ_ARR consists of at least one instance of SPK_WORDS, but could also exist of two or more instances. The WORD_SEQ either just consists of a sequence of words enclosed in quotation marks or could even contain plausibility values like shown in TABLE V.

TABLE V. UTTERANCE WITH PLAUSIBILITY VALUES

```
{ "recog_utter":  [
   {"v": "tarom", "p": "0.85"},
   {"v": "three four one", "p": "0.4"},
   {"v": "x-ray", "p": "0.25" }
  ]}
```

WORD_SEQ = "WORD WORD* " | WORDS_PLAUS_ARR
WORDS_PLAUS_ARR = [VAL_PLAUS, VAL_PLAUS* …] |
WORDS_PLAUS_SEQ | WORDS_PLAUS_SEQ*
VAL_PLAUS = {"v": "WORD WORD*", "p": PLAUS}
PLAUS = -1 | 0.0, …. 1.0

"PLAUS" could take a value of -1, meaning that plausibility is not defined by the given implementation, or a value between 0.0 and 1.0. The higher the plausibility value, the more certain the Speech-to-Text recognition is that the corresponding word sequence is correct. This has the consequence that alternative word sequence outputs must also be possible as shown in TABLE VI.

TABLE VI. SPEECH-TO-TEXT ALTERNATIVES WITH PLAUSIBILITIES

```
{
"recog_utter":  [
  {"wordsProb": [ {"v": "lufthansa", "p": "0.8"} ]},
  {"wordsProb": [ {"v": "papa", "p": "0.85"}, {"v": "tango", "p": "0.6"} ]},
  {"wordsProb": [ {"v": "buy", "p": "0.8"}, {"v": "bye", "p": "0.1"} ]}
 ]
}
```

WORDS_PLAUS_SEQ = {"wordsProb": WORDS_PLAUS_ARR}

Here, the Speech-to-Text recognition has returned four different outputs "lufthansa papa buy", "lufthansa tango buy", "lufthansa papa bye", and "lufthansa tango bye" with different plausibilites.

The contents of TABLE V could also be written as shown in table TABLE VII. Here, we just have one word sequence "tarom three four one x-ray", but with different plausibilities for each of the word sequences.

TABLE VII. ONE ALTERNATIVE WITH PLAUSIBILITIES

```
{ "recog_utter":  [
  {"wordsProb": [ {"v": "tarom", "p": "0.85"} ]},
  {"wordsProb": [ {"v": "three four one", "p": "0.4"}]},
   {"wordsProb": [ {"v": "x-ray", "p": "0.25"}]}
] }
```

B. *JSON format for the annotations*

In addition to the format for Speech-to-Text recognitions a JSON based format for ontology instructions (annotations) is defined as well. TABLE VIII contains a simple example for this definition with the annotations for just one speaker.

TABLE VIII. ANNOTATIONS FOR ONE SPEAKER

```
{
"filename": "2020-11-20__10-39-13-08.wav",
  "commands": [
    {"csgn": "BRU880", "type": "CONTACT",
          "valu": "BRATISLAVA_RADAR"},
    {"csgn": "BRU880", "type": "CONTACT_FREQUENCY",
          "valu": "134.475"},
    {"csgn": "BRU880", "type": "FAREWELL"}
  ]
}
```

ANNO_FILE = FILE_DESC ALL_CMDS
ALL_CMDS = CMDS1 | CMDS_MULT_SP
CMDS1 = "commands": [KEYW_SEQ, KEYW_SEQ,*]
CMDS_MULT_SP = "abstraction_layer_ontology_command":
[SP_CMDS SP_CMDS*]
SP_CMDS = { "speaker": SPK, CMD1 }
KEYW_SEQ = KEYW_VALUE, KEYW_VALUE*
KEYW_VALUE = KEYW VALUE
VALUE = "value" |VAL_PLAUS_A
VAL_PLAUS_A = {"v": "value", "p": PLAUS}
KEYW = "csgn" | "reas" | "type" | "valu" | "unit" | "qual" | "cond"
SPK, FILE_DESC, PLAUS already defined in subsection V.A

TABLE IX illustrates an example containing the initial call of the pilot and the resulting ATCo commands, in which the VALUE of DIRECT_TO command contains a plausibility value.

TABLE IX. ANNOTATIONS FOR MULTIPLE SPEAKERS


```
"abstraction_layer_ontology_command": [ {
  "speaker": "Pilot",
  "commands": [
    {"csgn": "GAC435F", "reas": "REPORTING", "type": "DESCEND",
            "valu": "170", "unit": "none"},
    {"csgn": "GAC435F", "reas": "REPORTING", "type": "DIRECT_TO",
            "valu": "WW973", "qual": "none"}
  ]},   {
  "speaker": "ATCO",
  "commands": [
    {"csgn": "GAC435F", "type": "DIRECT_TO", "qual": "none",
            {"v": " WW972", "p": "0.8"}},
    {"csgn": "GAC435F", "type": "NO_SPEED_RESTRICTIONS"}
      ]}
]
```


C. *JSON format for transcriptions and annotations and multiple hypotheses*

In some cases, like for unsupervised learning, it makes sense to have a format which contains the output of the transcription and the corresponding annotations. TABLE X shows an example for this scenario.

TABLE X. TRANSCRIPTION, CLASSIFICATION, AND ANNOTATION


```
{
"filename": "2020-11-20__09-39-35-34.wav",
"abstraction_layer_word_sequence": [
  { "recog_speaker": "ATCO", "recog_utter":
          "seven two five contact graz on one one nine three bye bye" }
  ],
"abstraction_layer_classific_seq": [
    { "recog_speaker": "ATCO", "recog_utter":
          "csgn csgn csgn type valu unkn vare vare vare vare type type" }
  ],
"commands": [
  {"csgn": "IGA725", "type": "CONTACT", "valu": "GRAZ_RADAR"},
  {"csgn": "IGA725", "type": … "valu": "119.300"},
  {"csgn": "IGA725", "type": "FAREWELL"}
  ]
}
```


TRANS_ANNO_FILE = FILE_DESC TRANS_CL_ANNO
TRANS_CL_ANNO = TRANS_ANNO | TRANS_CLAS_ANNO
TRANS_ANNO = TRANS ALL_CMDS
TRANS_CLAS_ANNO = TRANS CLAS ALL_CMDS
TRANS = "abstraction_layer_word_sequence": UTTERANCE
CLAS = "abstraction_layer_classific_seq": CLASSIFICATION
FILE_DESC, UTTERANCE already defined in subsection V.A,
ALL_CMDS already defined in subsection V.B.

CLASSIFICATION represents the classification of the UTTERANCE. It contains exactly one classification type for each word in the transcription. Allowed words here are the same set of words as defined for KEYW (defined in subsection V.B), and. “vare” (value rest), “grtg” (greeting), “spea” (speaker) and “unkn” (unknown).

The output of the automatic annotation is not always unique as shown by the examples in the previous sections. We have already shown alternative output on transcription level.

TABLE XI. TWO HYPOTHESES WITH DIFFERENT SPEAKERS


```
{"filename": "2020-11-20__09-39-35-34.wav",
"hypotheses": [{
  "abstraction_layer_word_sequence":
          "Pilot: good morning ... ATCO: qatari seven zero ...",
  "abstraction_layer_ontology_command": [{
     "speaker": "Pilot",
     "commands": [{"csgn": "DLH123",
          "type": {"v": "NO_CONCEPT", "p": "0.5"} }
          ]},
{
     "speaker": "ATCO",
     "commands": [{"csgn": "QTR703A", "type": "ALTITUDE"…}]
 } ]  },
{
  "abstraction_layer_word_sequence":
          "qatari seven zero three alfa correct seven thousand",
 "abstraction_layer_classific_seq":
          "csgn csgn csgn csgn valu type valu valu",
  "abstraction_layer_ontology_command": [{
       "speaker": "ATCO",
       "commands": [{"csgn": "QTR70", "type": "INFORMATION",
                "sndt": "ATIS","valu": "A" }]
       }]
  }]
}
```


VI. EXPERIMENTAL SETUP

Experiments to demonstrate the current capabilities of the current automatic command extraction (see section IV) were conducted. The dataset used to conduct the experiment was obtained from the HAAWAII project for London airspace (from NATS air navigation service provider). The data from the London airspace consists of the TMA South sector and the Heathrow approach sector. The dataset comprised of 1,216 commands from 746 utterances, composed of both ATCo and pilot speakers. The number of commands in each utterance ranged between 1 and 7.

The quality of command extraction is evaluated by comparing the automatically extracted annotations to the gold annotations. Gold annotations refer to the annotations which are manually verified and corrected by a human expert. The metrics used to evaluate the quality of the extracted annotations are: recognition rate (RecR), error rate (ErrR), and rejection rate (RejR). The recognition rate (RecR) is defined as the number of correctly recognized commands divided by the total number of actually given commands. Error rate (ErrR) is the percentage of wrongly extracted commands, i.e., the number of commands extracted wrongly divided by the total number of commands actually given. Rejection rate (RejR) is the percentage of gold annotations which are not extracted. A wrong command extraction is considered as a rejection, if for a given gold annotation (i) nothing is extracted, or (ii) command type NO_CONCEPT is extracted, or (iii) the correct command type is extracted, but with the callsign NO_CALLSIGN and this differs from the given gold annotation.

For a given speech utterance, each instruction (see Fig. 1) is treated as one big word. The Levenshtein distance between the gold annotation and the results of command extraction is then calculated, which returns the number of substitutions (subs), insertions (ins), and deletions (del). TABLE XII gives an overview about the different metrics and illustrates an example how they are calculated. In the table #gold defines the total number of commands in the gold annotation. #match defines the number of matches, which is #gold – subs – del.

TABLE XII. METRIC DEFINITION

Metric	Calculation
Command Recognition Rate (**RecR**)	**RecR** = #matches / #gold
Command Recognition Error Rate (**ErrR**)	**ErrR** = (subs + ins) / #gold
Command Rejection Rate (**RejR**)	**RejR** = del / #gold
Callsign Recognition Rate (**CaR**)	Same as RecR but only for callsigns without instructions
Callsign Recognition Error Rate (**CaE**)	Same as ErrR, but only for callsigns without instructions
Callsign Rejection Rate (**CaRj**)	Same as RejR, but only for callsigns without instructions
If the command extraction results in different callsigns, the calculation is done for each callsign. See example below, which also illustrates that the sum of RecR, ErrR, and RejR can exceed 100%.	
Example	
Gold Annotation	**Command Extraction**
AFR123 INIT_RESPONSE AFR123 TURN LEFT AUA1AB SPEED 140 kt DLH123_NO_CONCEPT	AFR123 DIRECT_TO OKG none AFR123 INIT_RESPONSE AFR123 TURN RIGHT AUA1AB NO_CONCEPT DLH123 NO_CONCEPT
Result:	
RecR = 2/4 = 50% (green)	*ErrR = 2 / 4 = 50% (purple)* / *RejR =1/4 = 25% (yellow)*

If the results of command extraction contain either NO_CONCEPT or NO_CALLSIGN, these substitutions and insertions are always calculated as deletions, i.e., these extractions contribute to the rejection rate and not to the error rate (as shown in the example in TABLE XII).

TABLE XIII. EXAMPLE OF METRIC CALCULATION WITH INIT_RESPONSE AND SPEED COMMANDS SWITCHED OFF

Gold Annotation	Command Extraction
AFR123 TURN LEFT AUA1AB NO_CONCEPT DLH123 NO_CONCEPT	AFR123 DIRECT_TO OKG none AFR123 TURN RIGHT AUA1AB NO_CONCEPT DLH123 NO_CONCEPT
AFR123 INIT_RESPONSE is mapped to AFR123 NO_CONCEPT. However, both gold annotation and command extraction contain another command for AFR123. NO_CONCEPT is only added, if it is the only command, which is the case for AUA1AB with SPEED mapped to NO_CONCEPT.	
Result:	
RecR = 2/3 = 67% (green)	*ErrR = 2/3 = 67% (purple)* / *RejR = 0 = 0% (yellow)*

For calculation of the callsign rates CaR, CaE, and CaRj we just compare the callsigns from the gold annotation with the callsigns of the automatic extraction. For each utterance we consider the callsign only once, except when multiple callsigns are annotated or extracted. For the example in TABLE XII, this results in the three annotated and extracted callsigns AFR123, AUA1AB, and DLH123. As the ontology is still evolving, the annotations and extractions for different data sets are based on different versions of the ontology. In most cases, new ontology versions introduce new command types. The metric calculation has to take this into account so that older data sets can also be reused. If some command types were not considered in the gold annotation or by the extraction (set via a configuration file), these command types are deleted from both the annotation and from the extraction. Following the deletions, if the set of extracted annotations for a callsign is empty, the command type NO_CONCEPT is added for this callsign instead. If INIT_RESPONSE and SPEED command types are not supported for the above example from the metric definition this would lead to the following result as shown in TABLE XIII.

VII. RESULTS

The results of command extraction for both ATCo and pilot utterances for the manually verified directories are shown below in TABLE XIV.

TABLE XIV. COMMAND EXTRACTION RESULTS

Directories	*#Utterances*	*# Cmds*	*RecR*	*ErrR*	*RejR*
DAY1 - SECTOR1 (ATCo)	217	330	97.3%	0.9%	1.8%
DAY1 - SECTOR1 (PILOT)	229	363	94.8%	1.1%	4.4%
DAY2 - SECTOR2 (ATCo)	128	226	96.5%	0%	3.5%
DAY2 - SECTOR2 (PILOT)	172	297	93.9%	0.3%	5.7%
All	746	1216	95.6%	0.6%	3.8%

From TABLE XIV we observe that the recognition rates are better and the error rates are lower for ATCo utterances as compared to pilot utterances. This is because ATCos are observed to adhere to the standard ATC phraseology more often than pilots. Pilots, on the other hand tend to use short phrases while reading back to the ATCos. Many times, pilots just read back numerical command values without mentioning the command type. For example, in the given pilot's readback utterance - *"one zero zero speed bird four seven five"*, *"one zero zero"* is the command value, which is a valid value for ALTITUDE, HEADING, or SPEED commands and the word "speed" belonging to callsign does help to reduce ambiguity.

TABLE XV illustrates the evaluation results when commands with just the numerical command value specified in the utterance are not extracted by our command extraction, i.e., line 27 in the algorithm shown in Fig. 2 would be missing. Whenever there is a difference to the rates presented in TABLE XIV the second number in the cell shows in bold which rate was achieved before.

TABLE XV. COMMAND EXTRACTION RESULTS WHEN SHEER NUMERICAL VALUES ARE NOT EXTRACTED

Directories	*#Utterances*	*# Cmds*	*RecR*	*ErrR*	*RejR*
DAY1 - SECTOR1 (ATCo)	217	330	97.3%	0.9%	1.8%
DAY1 - SECTOR1 (PILOT)	229	363	91.5% **94.8%**	1.1%	7.7% **4.4%**
DAY2 - SECTOR2 (ATCo)	128	226	96.5%	0%	3.5%
DAY2 - SECTOR2 (PILOT)	172	297	90.9% **93.9%**	0.3%	8.6% **5.7%**
All	746	1216	94.1% 95.6%	0.6%	5.5% 3.8%

The second number shows the result from TABLE XIV, when there is a difference.

It is important to note that extraction of commands when just the numerical command values are specified is implemented for just pilot utterances. The reason for this is, in addition to clearances, ATCos also give information regarding traffic and other miscellaneous information which contain numerical values. These values should not be wrongly extracted as command types. For example, in the utterance - *"easy eight two nine golf passing nine zero i can give you a turn north"*, the ATCo implies that he/she would give a TURN NORTH clearance once the aircraft passes an altitude of FL 90. The ATCo does not currently give any clearance here. The heuristic to extracting full commands from just numbers could wrongly extract value 90 as an ALTITUDE command. Moreover, ATCos seldomly use sheer numbers to give command clearances. Hence the minimal improvement in the recognition rates does not justify the increase in the error rates for ATCos.

From TABLE XV we observe that by not extracting commands using just their numerical values, the recognition rates go down significantly from 94.8% to 91.5% and 93.9% to 90.9% for the pilot utterances of DAY1 SECTOR1 and DAY2 SECTOR2, respectively. On the other hand, the error rates remain the same for both directories, which is why it works well in improving the overall quality of command extraction.

Using context information, i.e., using the surveillance data to predict, which callsigns are possible, also significantly improves the quality of command extraction. TABLE XVI shows the results of command extraction without using the callsigns from context. The second number in the cell is again a repetition of the numbers in TABLE XIV.

TABLE XVI. COMMAND EXTRACTION RESULTS WHEN CONTEXT INFORMATION IS NOT USED

Directories	*#Utterances*	*# Cmds*	*RecR*	*ErrR*	*RejR*
DAY1 - SECTOR1 (ATCo)	217	330	79.4% **97.3%**	7.9% **0.9%**	13.3% **1.8%**
DAY1 - SECTOR1 (PILOT)	229	363	72.7% **94.8%**	11.8% **1.1%**	16.8% **4.4%**
DAY2 - SECTOR2 (ATCo)	128	226	81.9% **96.5%**	13.7% **0%**	4.4% **3.5%**
DAY2 - SECTOR2 (PILOT)	172	297	72.1% **93.9%**	16.5% **0.3%**	16.5% **5.7%**
All	746	1216	76.5% **95.6%**	12.5% **0.6%**	12.8% **3.8%**

The second number shows the result from TABLE XIV, when there is a difference.

From TABLE XIV and TABLE XVI, we observe that the recognition rates fall drastically when context information is not used to extract callsigns. There is an overall decrease in the recognition rates from 95.6% to 76.5%. Moreover, error rates also increase significantly from 0.6% to 12.5% when context information is not used by the command extraction model. This is because both ATCos and pilots most often do not specify full callsign in their utterances. For example, BAW515 would not always be said as *"speed bird five one five"*, but *"speed bird one five"* or *"five one five"* could also be said quite commonly. Using context information not only decreases the error rates but also improves recognition rates, not just on callsign extraction level but also on the command level.

VIII. CONCLUSION AND OUTLOOK

Understanding ATC speech communications is so much more than just transforming speech into text. The ontology for ATC instructions shows that it is possible to put the content of such communications in a more standardized format.

This paper presented an extension of the original ontology, which makes it also usable for pilot utterances and therefore allows much more complex applications, which require a comparison between ATCo and pilot instructions. The here presented algorithm classifies each word of a text sequence, transforms the text into an ontology conform instruction and presents also a plausibility measurement to report the reliability of certain extracted elements. The defined JSON format allows a consistent exchange concerning Speech-to-Text transformation, ontology information or both together between different systems and applications. The format is by definition machine readable and easy to expand with additional key-value pairs while ensuring compatibility with old data.

The first results presented from the experiments for command extraction show already a quite impressive performance for ATCo and pilot utterances. With error rates for ATCos from 0% to 0.9% and 0.3% to 1.1% for pilots the algorithm can be considered quite reliable. Nevertheless, the approach still has to be tested on bigger and different data sets and improvements are still expected in the future as multiple projects which are working on the command extraction algorithm are still running. With improving recognition and error performances on ontology level complex applications seem to become possible. The abstraction of ATC communications allows a comparison of ATCo and pilot utterances and enables applications such as readback error detection to improve safety in the future.

ACKNOWLEDGMENT

Three projects of SESAR2020 industrial research and exploratory research have received funding from the SESAR Joint Undertaking under the European Union's grant agreement No. 734141, 874464, 874470 and 884287. The projects are named PJ.16-04-W1 (CWP HMI), PJ.10-96-W2 (HMI Interaction modes for Approach control), PJ.05-97-W2 (HMI Interaction Modes for Airport Tower), and HAAWAII (exploratory research). The project STARFiSH is funded by the German Federal Ministry of Education and Research.

REFERENCES

[1] 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.

[2] 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.

[3] HAAWAII homepage: www.haawaii-project.de, Highly Automatic Air Traffic Controller Workstation with Artificial Intelligence Integration, n.d.

[4] STARFiSH, research project funded by the German Federal Ministry of Education and Research, see for further information https://www.softwaresysteme.pt-dlr.de/de/ki-in-der-praxis.php, in German, n.d.

[5] PJ.05-97-W2 SESAR2020 funded industrial research projects under the European Union's grant agreement 874464, see for further information https://www.remote-tower.eu/wp/?page_id=888, and

https://www.remote-tower.eu/wp/?p=824 and https://www.sesarju.eu/index.php/projects/DTT, n.d.
[6] PJ.10-96-W2: SESAR2020 funded industrial research projects under the European Union's grant agreement 874470, https://cordis.europa.eu/programme/id/H2020_SESAR-IR-VLD-WAVE2-10-2019/de, n.d.
[7] K. Hofbauer and St. Petrik, "ATCoSIM Air Traffic Control Simulation Speech Corpus," Technical Report, May 2008, TR TUG-SPSC-2007-11.
[8] 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.
[9] 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.
[10] 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.
[11] http://www.json.org, JSON = JavaScript Object Notation, n.d
[12] AcListant homepage: www.AcListant.de, AcListant = Active Listening Assistant, n.d.
[13] A. Schmidt, "Integrating situational context information into an online ASR system for Air Traffic Control," Master Thesis, Saarland University (UdS), 2014.
[14] 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.
[15] ATCO2 project homepage: https://www.atco2.org/ ATCO2 = Automatic collection and processing of voice data from air-traffic communications, n.d.
[16] Flight Safety Foundation, FSF ALAR Briefing Note, 2.3 – Pilot-Controller Communication," 2000.
[17] K. Cardosi, "An analysis of En Route Controller-Pilot Voice Communications,", Tech. Rep. DOT/FAA/RD-93-11, 1993.
[18] K. Cardosi, "An Analysis of Tower (Local) Controller-Pilot Voice Communications," Tech. Rep. DOT/FAA/RD-94/15, 1994.
[19] S. Chen, H. Kopald, R.S. Chong, Y.-J. Wei, and Z. Levonian, "Read Back Error Detection using Automatic Speech Recognition", Twelfth USA/Europe Air Traffic Management Research and Development Seminar (ATMS2017), Seattle, WA, USA, 2017.
[20] G. van Es, "Air-ground communication safety study: an analysis of pilot-controller occurrences," EUROCONTROL, 2004.
[21] O.V. Prinzo, "The computation and effects of air traffic control message complexity and message length on pilot readback performance," in Proceedings of Measuring Behavior 2008, 6th International Conference on Methods and Techniques in Behavioral Research, Maastricht, The Netherlands, 2008.
[22] Airbus, "Flight Operations Briefing Notes, human Performance, Effective Pilot / Controller Communications," 2004.
[23] A. Isaac, "Effective Communication in the Aviation Environment: Work in Progress," *Hindsight*, 5, pp. 31–34, 2007.
[24] H. Helmke, M. Kleinert, O. Ohneiser, H. Ehr, S. Shetty, "Machine Learning of Air Traffic Controller Command Extraction Models for Speech Recognition Applications," IEEE/AIAA 39th Digital Avionics Systems Conference (DASC), Hosted virtually, 2020.