

ENHANCING COMMAND RECOGNITION IN AIR TRAFFIC CONTROL THROUGH ADVANCED CLASSIFICATION TECHNIQUES

Narayanan Srinivasan  0000-0003-1208-7302

S. R. Balasundaram  0000-0002-6590-5076

Department of Computer Applications, National Institute of Technology,
Tiruchirappalli, Tamil Nadu – 620 015, India

ABSTRACT

This paper addresses the persistent challenges in speech processing within the Air Traffic Control (ATC) domain, a field where despite extensive research, issues such as handling noisy environments, accented speech, and the need for strict adherence to standard phraseology continue to undermine conventional language models. Our study employs a hybrid approach that integrates syntactic analysis with advanced machine learning classification algorithms – Logistic Regression, Lagrangian Support Vector Machine, and Naïve Bayes. By mixing and matching algorithms tailored for specific aspects of speech processing, our approach moves away from traditional reliance on a singular integrated system, illustrating through rigorous testing with the ATCOSIM dataset that such a multifaceted strategy markedly improves command recognition accuracy and adapts more effectively to the unique linguistic features of ATC speech. Results highlight the superior performance of Logistic Regression across various command recognition categories, pointing towards a promising direction for future advancements in ATC speech recognition technologies aimed at reducing human workload and increasing automation precision. This paper explores the complexities of the required analysis techniques and underscores the necessity of employing diverse algorithms in the processing pipeline to enhance overall system accuracy.

Keywords

Air Traffic Speech; air traffic speech classification; Error Detection and Correction; air traffic command recognition.

1. INTRODUCTION

The use of speech in air traffic communication has been a significant topic stretching back to the 1970s. Miller [1] has defined the objective of speech recognition in electronic systems as being able to convert user speech into text and then use that text for various purposes like automation, workflow management, etc. For example, in Air Traffic Communication (ATC), the pilot in the aircraft cockpit receives instructions from a controller on the ground, who in turn receives acknowledgements, questions, approvals from the pilot. Speech recognition in the cockpit is seen as potentially more useful than in the controller environment – in other words, it is expected that controller commands to the pilot in the cockpit will be interpreted onboard, then the decoded speech will be used to automate certain actions, feeding into other avionics systems for implementing automation workflows. ATC speech recognition poses its own distinctive challenges, including noisy environments, heavily accented speech, rapid speech rates, communication ambiguities, potential misinterpretations of context, etc. Newberry [2] has discussed the application of speech in the cockpit and the integration of data link and Flight Management Systems (FMS). Ness and Herbert [3] have described a speech interface that gives directions to drivers, similarly to how controllers give directions to pilots.

More recently, Helmke et al. [4] have explained how vocabulary specific to local contexts affects the accuracy of ATC speech recognition. Chen et al. [5] have highlighted the importance of command recognition in detecting and analyzing read-back errors. They also note that in a 6-year dataset on runway incursions, there is an occurrence of one such incursion per 40.7 million commands; although the probability is low, the consequences of runway incursions are very costly. Cerna et al. [6], in turn, have described how the context of speech recognition changes with respect to different approach areas or controller working positions. Lin [7] discusses how commands are the best way to determine the intent of an instruction given by the controller to the pilot.

2. CHALLENGES IN COMPLETE UTTERANCE RECOGNITION IN ATC SPEECH RECOGNITION

Although there is an established standard phraseology for ATC commands, Helmke et al. [8] report that $\geq 25\%$ of spoken commands do not follow the standard phraseology. There is heavy mix of general English words that emerge naturally during conversation, and the use of local language dialects is another primary factor in this non-adherence to the standard phraseology. As an example, Berlin Air may be articulated in many ways – as *Berlin*, *Berlin Air*, *Air Berlin*. This adds additional complexity for any language model designed to perform linear matching of the

words. Training a language model to correctly recognize all such permutations of a n-gram sequence is still not an easy task.

Similar problems are posed by numerical expressions, which often create an ambiguous interpretation. For example, 4000 should be uttered as *four zero zero zero*, but local dialects might yield *four thousand* or *forty hundred* or *four triple zero*. Even though such variant ways of calling out numbers are against the standard, they are nevertheless common in ATC communication. Ambiguities also occur with numbers relating to altitude, speed and occasionally headings. For example, REDUCE BY 200 could ambiguously refer to altitude or airspeed, if not explicitly given as REDUCE SPEED or REDUCE ALTITUDE. Another common scenario arises when calling out the radio frequencies. If 19.1 is the frequency called out, it may be uttered as *one niner decimal one* or as *one niner point one*, which are both acceptable variants, but when called out non-standardly as *nineteen one*, this can lead to erroneous decoding.

Chen et al. [5] have reported on the presence of such errors in read-back scenarios, where pilots must repeat instructions back to controllers for verification. Holone and Nguyen [9], in turn, have noted that not infrequently controllers opt to communicate with local aircraft in their own language or greet another pilot in their common native language, rather than using English. In these kinds of scenarios, we cannot expect a confined language model to be able to cope with these discrepancies. Bakr [10] discusses similar communication challenges in marine traffic.

3. COMMAND RECOGNITION TECHNIQUES

Cerna et al. [6] introduced a method to integrate contextual information with ASR hypotheses using Levenshtein distance, successfully reducing the Command Error Rate. In our own earlier work [11], we found that string distance methods are less efficient in an ATC system and not applicable to a different corpus or a different context. Cerna et al. also speak about one more parameter, Concept Error Rate, which is extracted using sequence labelling approach. In our work, we have used a different approach to extract concepts, which significantly reduces both Concept Error Rate and Command Error rate. Helmke et al. [4] have also advocated for reducing CER using context-based rescoring of N-best lists, achieving a 30% reduction in Command Error Rate using a 4-hour specific ATC dataset. This, however, heavily relies on the decoded segments from the speech engine.

Chen et al. [5] classifies the ATC instructions into various categories, including Aircraft Identifier (ACID), Line up and Wait (LUAW), Cleared for Takeoff (CFT), Hold Short (HS), Cross (Cross), Turn (Turn), Taxi (Taxi), and Continue (Continue). They suggest using a rule-based classification technique to determine the recognition of these command categories. However, such techniques may not be of great help when an instruction contains multiple segments, potentially leading to a multi-class category rather than a simple command. Helmke et al. [4] use a similar

approach – semi-supervised learning from out-of-domain vocabulary – to improve concept recognition and command recognition. As noted above, due to the strict vocabulary and sentence formation, out-of-domain words are common in ATC conversation. When out-of-domain words are used, however, they can disrupt concept and command recognition.

The team at Airbus [12] carried out an analysis of how callsigns can be detected from ATC utterances, concluding that Neural Language and n-gram models did not show much difference in terms of accuracy, and also that corpus collection is still a challenge. Cerna et al. [6] used a hybrid acoustic modelling-based adaptation and then iterative methods to achieve a significant command and concept recognition rates from 70% to 90% under different conditions. Lastly, Lin [7] has confirmed the challenges involved in ATC speech decoding and emphasis on the Language Understanding (LU) step.

In summary, if the decoded utterance is to be used by various other systems for automation purposes, the concept and command recognition metrics are critical to their performance. In this study, we apply a machine learning based classification algorithm to derive commands and concepts from ATC utterances. With minimal supervised data, we show that meaningful results can be achieved for command recognition.

4. PROBLEM STATEMENT

Speech applications hold significant relevance across various sectors today, with Air Traffic communication (ATC) between pilot and Air Traffic Controller being no exception. A vital component in the overall ATC speech recognition process is improving error detection and correction mechanisms, which directly influence the accuracy of the speech output. Another critical measure of the effectiveness of ATC speech recognition is the Command Error Rate (CmdER), defined as the ratio of commands recognized correctly to the total number of commands in the domain. Previous studies, in our view, have treated CmdER as an auxiliary rather than a fundamental metric for evaluating speech recognition. Additionally, these studies did not employ machine learning algorithms for detailed classification to the lowest level. Without accurate command recognition, the decoded utterance may not be useful for any further applications in ATC speech recognition. The purpose of ATC speech recognition systems is multifold: not only to provide a textual representation so that pilots/ATCs can see what was spoken, but also to enhance automation opportunities and reduce the human workload in this domain. Achieving this requires high accuracy, which remains challenging due to the inherent complexities of the ATC environment.

5. ARCHITECTURE

The proposed system architecture comprises several processing modules, each designed to handle distinct aspects of speech recognition. The initial module in the process chain is the Syntax Separation Module, subsequently followed by different classification modules. The purpose of having multiple classification modules is to support different modules based on the context and use cases.

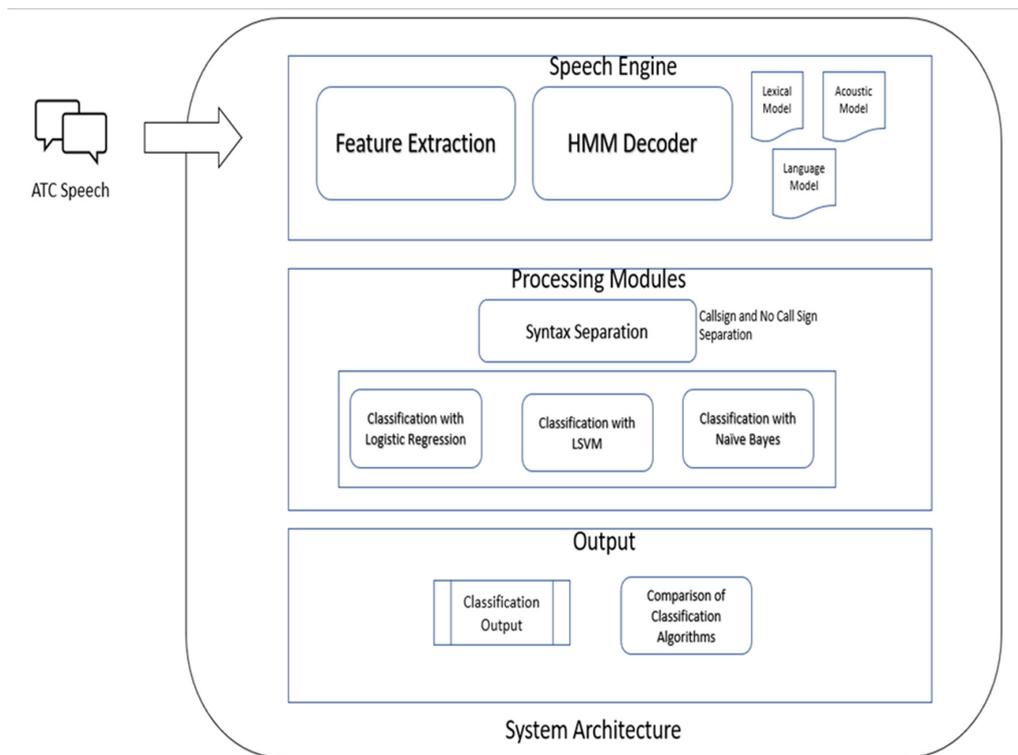


Fig. 1. System Architecture

6. METHODOLOGY

The methodology employed in this study utilizes the Sphinx speech recognition engine, as proposed by Lamere et al. [13], combined with the dataset developed by Hofbauer et al. [14]. This methodology was applied on the decoded output of the speech recognition engine as described in .

In the proposed architecture as in , Command Classification of the speech utterances is implemented using the following algorithms:

1. SyntaxPARSE (ST) – which separates the utterance into Callsign (CS) and Rest of CallSign (RCS)
2. Classify_CS (CS) using LR, LSVM, NB

3. Classify_RCS (RCS) using LR, LSVM, NB
4. Compare_Results (ST) – the main algorithm

where LR is Logistic Regression, LSVM is Lagrangian Support Vector Machine and NB is Naïve Bayes

Algorithm SyntaxPARSE (ST)

Input : ST – communication string

Output : CallSign part and Rest of Call sign part from the communication string (ST)

Begin

```

CS = Separate(ST,1) /*Separate callsign from ST
RCS = Separate(ST,2) /*Separate rest of callsign from ST
Return CS, RCS

```

End.

Algorithm Classify_CS (CS)

Input : CS – Callsign part of the communication string – output of SyntaxPARSE()

Output : Classifier of the CS string – Callsign (CS) or No Callsign (NCS)

Begin

```

Initialize (LR)
Initialize (LSVM)
Initialize (NB)
Set tempLR = Classify (LR, CS)
Set tempLSVM = Classify (LSVM, CS)
Set tempNB = Classify (NB, CS)
Return tempLR, tempLSVM, tempNB

```

End.

Algorithm Classify_RCS (RCS)

Input : RCS – Rest of Callsign part of the communication string – output of SyntaxPARSE()

Output : Classifier of the RCS string – Command Recognition Output (CLIMB, DECENT, TURN)

Begin

```

Initialize (LR)
Initialize (LSVM)
Initialize (NB)
Set tempLR = Classify (LR, RCS)
Set tempLSVM = Classify (LSVM, RCS)

```

```
    Set tempNB = Classify (NB, RCS)
    Return tempLR, tempLSVM, tempNB
End.
```

Algorithm CompareResults

Input : ST – Communication String
Output : Results of Callsign and Rest of callsign classification

```
Begin
    tempCS, tempRCS = SyntaxPARSE (ST)
    Print Classify_CS(tempCS)
    Print Classify_RCS(tempRCS)
End.
```

7. SIGNIFICANCE OF COMMAND RECOGNITION

Holone and Nyugen [9] have noted various challenges faced by speech recognition systems for ATC, which make achieving a robust speech recognition system with great accuracy is notably difficult. As noted above, these include heavily accented speech, local language usage despite a standard vocabulary, and radio channel noise. At the same time, dynamic word choices or sentence formation is highly restricted in ATC; the words and sentence structures used are largely fixed (referred to as the standard vocabulary and grammar). Chen et al. [5] detail a lengthy list of command types in ATC domain. In a typical situation, instructions are classified into the following categories:

1. Gate instructions
2. Taxi instructions
3. Runway instructions
4. In flight instructions

At larger airports, different controllers are designated for each category of instructions, while at smaller airports, the same controller may handle instructions in all categories. The instructions are fixed for a given airport – in other words, there are no dynamic instructions used either by the controller or pilot, except for the potential mix of local language and out-of-vocabulary words. The aircraft tail numbers or callsign will be different for each airport, but references to runways, taxiways, waypoints, altitudes, speeds are almost fixed for an airport and these words form part of airport vocabulary as well.

The complexity of instructions received by an aircraft varies. For example, “*Lufthansa Two Five Eight Push Back*” is a simple instruction given in Gate category. “*British Five Two Six Seven Cleared for Takeoff from Runway 08 RIGHT*” is another, medium-complex example from the Runway instructions category. A complex

instruction in ATC involves one or more commands. For example, “*United Seven Six Five Two Take Taxiway D and exit runway 54 then cross Taxiway B and runway 32*” is a complex instruction.

An ATC instruction can be syntactically defined as follows:

<CALLSIGN> <INSTRUCTION> <INSTRUCTION DATA>

In other words, there are the 3 critical pieces of information that need to be decoded correctly from an instruction. The <INSTRUCTION> itself can be further classified based on the above four categories. <INSTRUCTION DATA> is a composite utterance where each word must be decoded correctly. As we have noted, the not infrequent use of local vocabulary and words outside the standard vocabulary complicates the decoding process, making it challenging for the speech engine to accurately extract these three critical pieces of information. Next, interpretation of the decoded utterance involves the need to parse the instruction further to separate the critical and non-critical information from the utterance. The Hidden Markov Model (HMM) traditionally used for speech decoding may not include all combinations of words from the vocabulary and words out-of-vocabulary. Additionally, the presence of heavy accents can further obscure the clarity of the utterance, increasing the complexity of the task.

The Command Recognition approach simplifies the decoding problem by focusing on interpreting the command class rather than the complete command. For example, the instruction “*Lufthansa Eight Two Five three Push back*” can be segmented into <Lufthansa Eight Two Five Three> <Push Back>. This method primarily seeks to identify the command “*Push Back*”. While an HMM model would individually recognize “*Push*” and “*Back*”, a 2-gram language model would recognize the instruction is “push back”. However, if the uttered command deviates, appearing as something like “*Pusk Bac*”, the language model may fail to recognize this as “push back”. Command recognition can determine this utterance as “push back” easily. A classification model, trained with different combination of the utterance “push back,” is used to determine the command classification.

As stated above, training a language model for different combinations is a challenging task, and then normalizing the prediction to choose a specific n-gram is yet another challenge. Classification models, on the other hand, are trained to predict a specific command. To train the classification model for a command category, we have used different combinations of utterances for the same command. Experiments were conducted using the ATCOSIM dataset prepared by Hofbauer et al. [14]. On the other hand, if only the 3 critical pieces of information must be decoded, this can be done using the command recognition approach. The proposed algorithm first uses syntactic analysis to separate out the 3 critical parts of the utterance. Then each part is then run through a customized classification model to determine its class. A n-gram language model used by the speech engine tries to predict the word combination based on the training data. A classification model

predicts the class of the word rather than the exact word that was spoken. This is a key difference in applying the classification model for this problem.

8. BUILDING COMMAND RECOGNITION MODELS

A significant hurdle in developing command recognition models is collecting and compiling adequate training data for the classification model. Lin [7] highlights that this is one of the challenging tasks in tailoring language models for specific airport communications. Airports maintain comprehensive databases of aeronautical charts, known as Airport Information or Airport Map Databases. These databases, available in digital formats, contain detailed information about airport layouts including taxiways, runways, gate numbers, waypoints, and procedures for arrivals and departures, all in text format.

To generate the training data necessary for building effective classification models, a tool or automated system capable of producing the possible combinations of commands and instructions specific to an airport can be utilized. This system converts the detailed text information from the Airport Map databases into formatted text that can serve as input for training language models. With minimal real-time audio data from the airport, this approach allows for the construction of a classification model with reasonable accuracy. The performance and accuracy results of these models will be presented below.

Furthermore, adapting the language model to accommodate the details of different airports involves incorporating additional corpora to account for local variations in speech and terminology. For our research, the ATCOSIM dataset was instrumental as it includes a variety of audio samples along with their manually decoded texts. We conducted experiments using audio from a single male speaker specific to one airport to develop and refine our classification model.

9. COMMAND RECOGNITION CLASSIFICATION

Classification algorithms are supervised machine learning algorithms used to predict categorical values based on training data. The classification can result in multi-class output for the given set of inputs. A discrete output function $f(y)$ is mapped to an input variable x in a classification algorithm. The following algorithms were used in the experiments:

1. Logistic Regression
2. Lagrangian Support Vector Machine
3. Multinomial Naïve Bayes

Classification algorithms were used in two applications in this experiment. As stated earlier, syntactic analysis was also used prior to applying the classification

algorithm. The output utterance from the speech engine output was first split based on the syntax, as follows:

<CALLSIGN> <REST OF CALLSIGN>

A rule-based algorithm was written to split the utterance into two categories – “callsign” and “rest of callsign.” We applied classification to determine if the “callsign” part of the split can be classified as a “callsign” or “not a callsign.” As can be observed in the output of the speech engine, a callsign cannot be determined in several cases. There are multiple reasons for this: additional words being added to the instruction, words trimmed at the beginning or end of the callsign, noise issues causing loss of data critical to identify the callsign, etc. Delpech et al. [12] at Airbus performed various experiments to extract callsign from spoken instructions. In the present study, we did not attempt to extract the callsign, only to determine if we have sufficient information to determine the callsign using the classification algorithm. A typical callsign should consist of two aviation words, followed by 3 or 4 numbers uttered as aviation numerals – for instance “AERO LLOYD FIVE FIVE FIVE” or “ALITALIA FOUR FOUR SEVEN ONE”. Examples of invalid callsigns are “ALITALIA FOUR FOUR ALSO IDENTIFIED”, “ALITALIA ONE ONE FIVE AIR”.

The second part of the utterance (the “rest of callsign”) is classified into multiple categories. The categories are “CLIMB,” “CONTACT,” “DESCEND,” “DIRECTTO,” “PROCEEDDIRECT,” “PROCEED,” “RHEIN,” “RHEINRADAR,” “SETCOURSE,” “SET,” “TURN,” “TURNHEADING,” “TURNLEFT,” “TURNRIGHT”. These categories are much more detailed than the classification used by Chen et al. (2017). Classification of the rest of the callsign is especially important because the subsequent words in the utterance can be interpreted only if this classification is correct. For example, “CLIMB FLIGHT LEVEL THREE HUNDRED.” If this instruction is classified into “CLIMB” category, then Flight Level Three hundred can be interpreted as the aircraft being instructed to climb to 30,000 feet. In cases like “CONTACT AH LEFT BY TEN DEGREES,” however, there is a certain ambiguity. Is this to be classified as a “CONTACT” instruction or a “TURN LEFT” instruction? If it is classified as “CONTACT,” then rest of the utterance does not help to interpret the statement correctly, whereas if it gets classified as “TURN LEFT,” then the rest of the utterance does indeed help in its full interpretation. This illustrates the importance of the correct classification of the “rest of callsign” part of the instruction. A language model may try to pick up the right words from the n-gram based on probability, but it cannot determine the interpretability of the instruction as a whole, whereas a classification algorithm helps to determine the interpretability of the instruction.

To the best of our knowledge, this conclusion on the application of classification algorithms to enhance the interpretability of ATC instructions has not previously been drawn by other researchers in the field.

10. COMMAND ERROR RATE (CMER) PERFORMANCE WITH DIFFERENT CLASSIFICATION ALGORITHMS

The classification algorithms were used for two categories of classifications:

1. Callsign or No Callsign classification (binary classifier)
2. Rest of callsign – Instruction type classification (multi classifier)

10.1 Callsign Classification

A callsign or no callsign classifier model was built using 27% of training data and evaluated using the remaining 73% as test data. Different variations of the same callsign were specifically included in the training to improve the accuracy of the classifier. 70% of the training data was labelled as “callsign” and 30% as “nocallsign.” The general classifier model metrics are as follows.

10.1.1 Naïve Bayes Classification

Naïve Bayes performed decently in determining if the utterance is a callsign or not. However, it could not differentiate if there is sufficient information in the utterance to determine if it is a callsign or not. Lesser accuracy may be attributed to this reason. For example:

“ONE ONE”
“AERO AERO”
“AIR AIR”

The above instructions are classified as “CALLSIGN” by Naïve Bayes, which is incorrect. At the same time, the algorithm determined the following as “NO CALLSIGN,” which is better.

“AFFIRM HOW ARE HOW ONE”
“AIR MALTA SOON AS SOON AS FOUR FOUR YOU’RE IN”
“BACK BACK”

Another interesting observation is that the following utterances, with certain omissions, were nevertheless correctly classified as “CALLSIGN”:

“BRITISH S= SIX SIX ZERO”
“FO= FOUR FOUR ONE SIX”

If rule-based logic is applied for the above scenarios, it may not clearly filter this out as “no callsign” instructions. But as classification is used, the algorithm can consider the omitted characters and still classify it as “callsign”. Thus, the classification helps in improving the overall accuracy. The detailed metrics of Naïve Bayes algorithm is listed in Table 1.

Table 1: Naïve Bayes Classification Results for CallSign

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| CALLSIGN | 0.72 | 1.00 | 0.84 | 34 |
| NOCALLSIGN | 0.00 | 0.00 | 0.00 | 13 |
| Accuracy | | | 0.72 | 47 |
| Macro avg | 0.36 | 0.5 | 0.42 | 47 |
| Weighted avg | 0.52 | 0.72 | 0.61 | 47 |

LSVM Classification

As shown in Table 2, the LSVM algorithm performed better than the rest of the algorithms in classifying the “callsign” or “nocallsign” classes.

Table 2: LSVM Results Sample

| | |
|----------------------------|-----------------------|
| AERO LLOYD FIVE FIVE FIVE | ["CALLSIGN"] |
| AERO LLOYD FIVE FIVE NINE | ["CALLSIGN"] |
| AERO LLOYD FIVE FIVE ONE | ["CALLSIGN"] |
| AERO LLOYD FIVE FIVE SEVEN | ["CALLSIGN"] |
| AERO LLOYD FIVE FIVE TWO | ["CALLSIGN"] |
| AERO LLOYD LLOYD | ["NOCALLSIGN"] |
| AERO LLOYD ONE ONE FOUR | ["CALLSIGN"] |
| AERO LLOYD TRY A TWO | ["CALLSIGN"] |
| AERO LLOYD TWO TWO SIX | ["CALLSIGN"] |

In Table 2, a few sample utterances are given. “AERO LLOYD FIVE FIVE NINE” is a callsign. The LSVM classification algorithm is able to classify variations of the utterances into “Callsign,” yet when critical pieces of information (the number part of the callsign) is missing, it is classified as “No Callsign”. This is illustrated in the bolded row of Table 2. The rule-based algorithm would not determine the variations as “Callsign,” and a classification algorithm performs better here. When compared with the other classification algorithm, LSVM worked best for the Callsign classification, with accuracy of 74%, which is the highest among the classification algorithms used in the experiment.

The following Table 3 lists the utterances which were classified as “No Callsign”.

Table 3: LSVM No Callsign samples

| | |
|-------------------------------|----------------|
| FOUR FOUR SINGA TWO | ["NOCALLSIGN"] |
| FOUR FOUR | ["NOCALLSIGN"] |
| FOX JET JET | ["NOCALLSIGN"] |
| FOX READ FIVE | ["NOCALLSIGN"] |
| FOX SWISSAIR SWISSAIR | ["NOCALLSIGN"] |
| FOXTROT CHECK RADIO CHECK JET | ["NOCALLSIGN"] |

The LSVM algorithm was able to very clearly classify the utterances in the above table as “No Callsign.” There is insufficient information to determine if the utterances can be classified as “Callsign”. The detailed metrics of LSVM algorithm is listed in the Table 4.

Table 4: LSVM Classification Results for Callsign and No Callsign

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| CALLSIGN | 0.75 | 0.97 | 0.85 | 34 |
| NOCALLSIGN | 0.67 | 0.15 | 0.25 | 13 |
| Accuracy | | | 0.74 | 37 |
| Macro avg | 0.71 | 0.56 | 0.55 | 47 |
| Weighted avg | 0.73 | 0.74 | 0.68 | 47 |

10.1.2 Logistic Regression Classification

Analysis of logistic regression yielded some interesting results. Sample output from logistic regression is listed in Table 5.

Table 5: Logistic Regression Classification Samples

| | |
|---|----------------|
| HEADING FLIGHT HUNDRED MILANO ONE ONE THREE FOUR DECIMAL FIVE TWO | ["NOCALLSIGN"] |
| HEADING OLYMPIC ONE ONE | ["NOCALLSIGN"] |
| HEADING ONE ONE ONE ONE | ["NOCALLSIGN"] |
| HEADING TWO TWO SIX FOUR | ["NOCALLSIGN"] |
| HEADING YOU CHARLIE PAPA PAPA | ["NOCALLSIGN"] |
| HIGHER HIGHER | ["NOCALLSIGN"] |
| HIGHER ONE ONE ONE ONE | ["NOCALLSIGN"] |
| HOTEL FIVE FIVE SINGA TWO | ["CALLSIGN"] |
| HOTEL HOTEL | ["CALLSIGN"] |

The first seven utterances are clearly “No Callsign” and the classification results are correct. Note the use of out-of-domain words like HIGHER in the conversation. HEADING clearly says this is not a callsign but belong to a different classification called “HEADING”. A callsign utterance cannot start with HEADING and the algorithm has correctly classified in these cases. At the same time, this classification did not yield better results, due to the classification errors as seen in the last utterances. Despite sufficient information not being available, this utterance has been classified as “CALLSIGN,” which is incorrect. The detailed metrics of Logistic Regression algorithm are listed in the Table 6.

Table 6: Logistic Regression Classification Results for Callsign and NoCallSign

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| CALLSIGN | 0.75 | 0.88 | 0.81 | 34 |
| NOCALLSIGN | 0.43 | 0.23 | 0.30 | 13 |
| Accuracy | | | 0.70 | 47 |
| Macro avg | 0.59 | 0.56 | 0.56 | 47 |
| Weighted avg | .66 | .70 | .67 | 47 |

Overall comparison of the classification algorithms.

Table 7: Comparison of Algorithms Callsign and NoCallSign



Overall comparison of the algorithms in terms of accuracy is given in Figure 4:

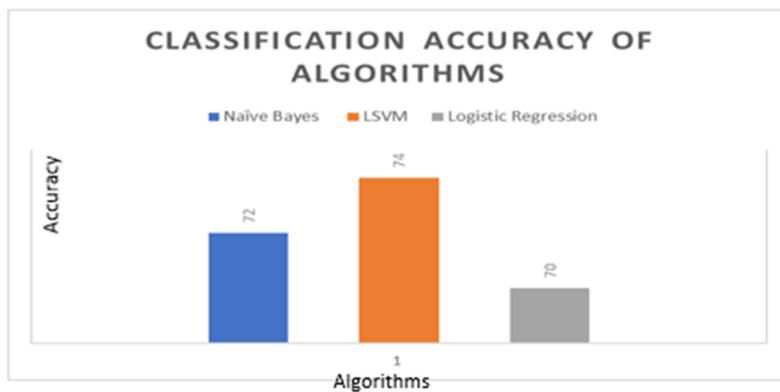


Fig. 4. Comparison of Algorithms

10.2 “Rest Of Callsign” Classification

As indicated earlier, rest of callsign was classified into multiple instruction types: “CLIMB,” “CONTACT,” “DESCEND,” “DIRECT TO,” “PROCEED DIRECT,” “PROCEED,” “RHEIN,” “RHEINRADAR,” “SET COURSE,” “SET,” “TURN,” “TURN HEADING,” “TURN LEFT,” and “TURN RIGHT.” As this is a multi-category classification, approximately 58% of the data was used to train and we evaluated the model on a different (fresh) data set, which is similar. Labels were given to ensure the classifier is determined with very few key words appearing in the training utterances. The general metrics of the model were as follows:

10.2.1 Naïve Bayes Classification

Naïve Bayes performed poorly for the Rest of Callsign classification with the tested data set. There were many incorrect classification outputs, as the algorithm could not use the specific clues from the utterance to classify it correctly. Table 8 lists the gross errors that are observed with this algorithm:

Table 8: Naive Bayes Rest of Callsign Samples

| | |
|---|---------------|
| DIRECT TO FRANKFURT | [“SETCOURSE”] |
| DIRECT TO GOTIL ONE THREE FOUR DECIMAL FIVE TWO | [“CONTACT”] |
| DIRECT TO NOVEMBER TANGO MIKE | [“SETCOURSE”] |
| DIRECT TO ONE QUITE DELTA | [“SETCOURSE”] |
| DIRECT TO TRASADINGEN | [“SETCOURSE”] |
| DIRECT TRASA | [“SETCOURSE”] |
| DIRECT TRASADINGEN | [“SETCOURSE”] |

The correct classification for the utterances listed above is “DIRECT TO”. At the same time, there are cases which the algorithm has classified utterances correctly, despite the clues not being so straightforward. Some examples are in Table 9 below:

Table 9: Naive Bayes Rest of Callsign Samples

| | |
|---|----------------|
| RHEIN IDENT YOU READ | [“RHEIN”] |
| RHEIN IDENTIFIED CLIMB FLIGHT LEVEL THREE YOU ARE YOU STILL | [“RHEINRADAR”] |
| RHEIN IDENTIFIED CLIMB FLIGHT LEVEL TWO NINE ALSO SET COSTA THANK YOU | [“RHEINRADAR”] |
| RHEIN IDENTIFIED CLIMB FLIGHT LEVEL TWO NINE ALSO | [“RHEINRADAR”] |
| RHEIN IDENTIFIED CLIMB FLIGHT ONE IDENTIFIED | [“RHEINRADAR”] |
| RHEIN IDENTIFIED CLIMB FLIGHT TO TWO SEVEN ZERO | [“RHEINRADAR”] |

Table 10, in turn, details the metrics for the different classifications which were evaluated.

Table 10: Naïve Bayes “Rest of Callsign” Results

| | precision | recall | f1-score | support |
|---------------|-----------|--------|----------|---------|
| CLIMB | 0.8 | 1 | 0.89 | 8 |
| CONTACT | 0.57 | 1 | 0.73 | 4 |
| DESCEND | 1 | 0.33 | 0.5 | 3 |
| DIRECTTO | 0 | 0 | 0 | 4 |
| PROCEEDDIRECT | 0 | 0 | 0 | 4 |
| PROCEED | 0 | 0 | 0 | 1 |
| RHEIN | 0 | 0 | 0 | 7 |
| RHEINRADAR | 0.67 | 0.93 | 0.78 | 15 |
| SETCOURSE | 0.25 | 1 | 0.4 | 3 |
| SET | 0 | 0 | 0 | 0 |
| TURN | 0 | 0 | 0 | 6 |
| TURNHEADING | 0 | 0 | 0 | 2 |
| TURNLEFT | 0.4 | 0.8 | 0.53 | 5 |
| TURNRIGHT | 0.8 | 1 | 0.89 | 4 |
| Micro avg | 0.58 | 0.58 | 0.58 | 66 |
| Macro avg | 0.32 | 0.43 | 0.34 | 66 |
| Weighted avg | 0.42 | 0.58 | 0.46 | 66 |

LSVM Classification

The LSVM algorithm showed a significant improvement in accuracy as compared to Naïve Bayes classification for the rest of Callsign utterances. The algorithm had given better results despite the number of words in the utterances. The algorithm was able to understand the clues for the utterances which are varying in length. A few examples are listed in Table 11.

Table 11: LSVM Rest of Callsign Samples

| | |
|--|-------------|
| DESCEND FLIGHT TWO ZERO EXPEDITE DESCENT AS CROSSING TRAFFIC | ["DESCEND"] |
| DESCEND FLIGHT LEVEL THREE EIGHT ZERO | ["DESCEND"] |
| DESCEND FLIGHT LEVEL THREE FO= SEVEN | ["DESCEND"] |
| DESCEND FLIGHT LEVEL THREE HUNDRED RATE ONE THOUSAND FIVE HUNDRED OR MORE | ["DESCEND"] |

| | |
|--|-------------|
| DESCEND FLIGHT LEVEL THREE ONE TWO FIVE | ["DESCEND"] |
| DESCEND FLIGHT LEVEL THREE ONE TWO | ["DESCEND"] |
| DESCEND FLIGHT LEVEL THREE ONE ZERO | ["DESCEND"] |
| DESCEND FLIGHT LEVEL THREE TWO ZERO RATE FIFTEEN HUNDRED OR MORE | ["DESCEND"] |
| DESCEND FLIGHT LEVEL THREE TWO ZERO | ["DESCEND"] |
| DESCEND FLIGHT LEVEL TO YOU ZERO SET COURSE TO FOXTROT | ["DESCEND"] |

Like the Callsign classification, the omission of characters did not impact the accuracy of LSVM algorithm. It performed well in cases where few characters are omitted in the decoded utterances. However, the algorithm had some challenges when the same utterance encompassed multiple commands. This is a normal scenario in ATC communication, as controllers will give a sequence of multiple instructions in the single communication to the pilots. The examples are given in the following Table 12:

Table 12: LSVM “Rest of Callsign” Samples

| | |
|---|---------------|
| CLIMB FLIGHT LEVEL THREE FIVE SINGA SET COURSE TO COURSE TO | ["SETCOURSE"] |
| CLIMB FLIGHT LEVEL THREE HUNDRED SET COURSE | ["SETCOURSE"] |
| CLIMB FLIGHT LEVEL TWO NINE ALSO SET COURSE TRASADINGEN | ["SETCOURSE"] |

Table 13 details the metrics for the different classifications which were evaluated with the LSVM algorithm.

Table 13: LSVM Rest of Callsign Results

| | precision | recall | f1-score | support |
|---------------|-----------|--------|----------|---------|
| CLIMB | 1 | 0.62 | 0.77 | 8 |
| CONTACT | 1 | 1 | 1 | 4 |
| DESCEND | 1 | 1 | 1 | 3 |
| DIRECTTO | 1 | 0.25 | 0.4 | 4 |
| PROCEEDDIRECT | 0.8 | 1 | 0.89 | 4 |
| PROCEED | 1 | 1 | 1 | 1 |
| RHEIN | 1 | 0.86 | 0.92 | 7 |
| RHEINRADAR | 0.94 | 1 | 0.97 | 15 |
| SETCOURSE | 0.38 | 1 | 0.55 | 3 |

| | | | | |
|--------------|------|------|------|----|
| SET | 0 | 0 | 0 | 0 |
| TURN | 0.8 | 0.67 | 0.73 | 6 |
| TURNHEADING | 0 | 0 | 0 | 2 |
| TURNLEFT | 0.83 | 1 | 0.91 | 5 |
| TURNRIGHT | 0.67 | 1 | 0.8 | 4 |
| micro avg | 0.83 | 0.83 | 0.83 | 66 |
| macro avg | 0.74 | 0.74 | 0.71 | 66 |
| weighted avg | 0.86 | 0.83 | 0.82 | 66 |

10.2.2 LOGISTIC REGRESSION CLASSIFICATION

Based on our experiments, the logistic regression algorithm yielded the best results, with accuracy of 84%. As compared with the LSVM algorithm, Table 14 lists the cases where this algorithm performed correctly.

Table 14: Logistic Regression “Rest of CallSign” Samples

| | |
|--|--------------|
| CLIMB FLIGHT LEVEL THREE HUNDRED SET COURSE | [“CLIMB”] |
| CLIMB FLIGHT LEVEL TWO NINE ALSO SET COURSE TRASADINGEN | [“CLIMB”] |
| CONTACT ~M YOU ONE ONE NATTENHEIM ONE ONE OF TWO | [“CONTACT”] |
| CONTACT ONE ONE ~M ONE AND | [“CONTACT”] |
| CONTACT RHEIN ONE THREE TWO DECIMAL FOUR ~S | [“CONTACT”] |
| DESCEND FLIGHT SET EIGHT ONE SIX LEFT | [“DESCEND”] |
| DIRECT TO FOXTROT ROMEO INDIA | [“DIRECTTO”] |
| RHEIN ~M MINUTE AND ~M ONE ONE ONE | [“RHEIN”] |
| RHEIN AH AND YOU ONE AH GUTEN | [“RHEIN”] |
| RHEIN AIR ONE THREE TWO DECIMAL FOUR | [“RHEIN”] |
| RHEIN ONE AND CALL | [“RHEIN”] |

Another interesting observation is that this algorithm performed well in cases where multiple instructions were present in the decoded utterance, which was not considered by the LSVM algorithm. Detailed metrics of Logistic Regression for different classifiers are given in Table 15.

Table 15: Logistic Regression Rest of Callsign Results

| | precision | recall | f1-score | support |
|---------------|-----------|--------|----------|---------|
| CLIMB | 1 | 0.62 | 0.77 | 8 |
| CONTACT | 1 | 1 | 1 | 4 |
| DESCEND | 1 | 1 | 1 | 3 |
| DIRECTTO | 1 | 0.5 | 0.67 | 4 |
| PROCEEDDIRECT | 0.8 | 1 | 0.89 | 4 |
| PROCEED | 1 | 1 | 1 | 1 |
| RHEIN | 1 | 0.86 | 0.92 | 7 |
| RHEINRADAR | 0.94 | 1 | 0.97 | 15 |
| SETCOURSE | 0.43 | 1 | 0.6 | 3 |
| SET | 0 | 0 | 0 | 0 |
| TURN | 0.8 | 0.67 | 0.73 | 6 |
| TURNHEADING | 0 | 0 | 0 | 2 |
| TURNLEFT | 0.71 | 1 | 0.83 | 5 |
| TURNRIGHT | 0.8 | 1 | 0.89 | 4 |
| micro avg | 0.85 | 0.85 | 0.85 | 66 |
| macro avg | 0.75 | 0.76 | 0.73 | 66 |
| weighted avg | 0.87 | 0.85 | 0.84 | 66 |

10.3 EVALUATION OF THE RESULTS (CALLSIGN)

Results for manual analysis for Callsign classification are given in Table 16.

Table 16: Comparison of Callsign, No Callsign

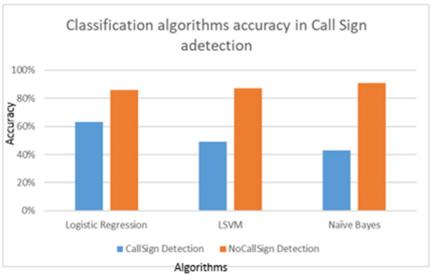
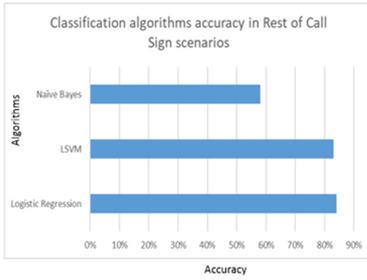
| Algorithm | Callsign Detection | No Callsign Detection | Chart Representation |
|---------------------|--------------------|-----------------------|--|
| Logistic Regression | 63% | 86% |  |
| LSVM | 49% | 87% | |
| Naïve Bayes | 43% | 91% | |

Fig. 5. Comparison of classification algorithms for callsign

10.4 EVALUATION OF RESULTS (REST OF CALLSIGN)

Table 17 lists the results of manual analysis of Rest of Callsign classification.

Table 17: Rest of Callsign – Analysis

| Algorithm | Multi Classifier Accuracy | Chart Representation |
|---------------------|---------------------------|---|
| Logistic Regression | 84% |  <p>Fig. 6. Comparison of classification algorithms for rest of callsign</p> |
| LSVM | 83% | |
| Naïve Bayes | 58% | |

The Naïve Bayes, Logistic Regression and LSVM algorithms were used to classify the ATC utterances and determine the commands. To our best of knowledge, such an attempt at achieving the best Concept Recognition (ConER) and Command Error Recognition (CmdER) using classification techniques has not been attempted earlier. Classification of Callsign or No Callsign and classification of Rest of Callsign can be directly attributed to Concept Recognition Rate.

This paper has admittedly discussed only first-level classifiers, but such classifiers are critical to determine the concept of the utterance. Therefore, the first level classifiers can be compared with the Concept Recognition rate from prior work; results are presented in Table 18.

Table 18: Summary of results comparison

| Work | Technique | Command Error Recognition Rate |
|--------------------|---|---|
| Helmke et al. [4] | SLM+Rescoring (N-best=5) | 16.5 |
| Helmke et al. [15] | ASR-SSA-C Concept | 35.9 |
| This work | Syntactic Separation for Callsign and Rest of Callsign and using ML classification algorithms | 16% (callsign best) 9% (rest of callsign best) |

Once the first-level classifiers are determined, predicting the rest of the information is much easier with rescored language models generated out of airport data.

10. CONCLUSIONS

Based on the results reported above, the Logistic Regression algorithm was shown to exhibit robust performance across all three categories of detection: Callsign Detection, No Callsign Detection and Rest of Callsign Detection. While other algorithms demonstrated better results in specific areas, Logistic Regression maintained consistent accuracy across the domain. This finding underscores the importance of understanding the details and dividing the processing pipeline to use different techniques to improve accuracy.

In this study, we used syntactical analysis along with classification algorithms, which are two different techniques in decoding speech. The integration of these methodologies in our analysis highlights the potential synergistic benefits of combining different techniques.

Looking ahead, our future work will focus on refining these algorithms to achieve higher accuracy levels. Additionally, we plan to incorporate more techniques into the processing pipeline to further improve accuracy. We firmly believe that a multi-faceted approach, utilizing diverse techniques, will produce superior results compared to relying solely on the performance of a speech engine. This strategy is crucial for advancing the field and developing more effective speech recognition systems in air traffic control environments.

REFERENCES

- [1] Miller C. U.S. Programme for Development of Satellite Services for Air Traffic Control. *J Navig.* 1990;43(1):26-31. <http://doi.org/10.1017/S0373463300013783>
- [2] Newbery RR. Integration of advanced displays, FMS, speech recognition and data link. *J Navig.* 1985;38(3):463-466. <http://doi.org/10.1017/S037346330003287>
- [3] Ness M, Herbert M. A route planning and driver information system for PLEIADES. *J Navig.* 1994;47(2):159-164. <http://doi.org/10.1017/S0373463300012078>
- [4] Helmke H, Klakow D, Motlicek P, Srinivasamurthy A, Szaszak G, Oualil Y. A context-aware speech recognition and understanding system for air traffic control domain. 2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU). <http://doi.org/10.1109/ASRU.2017.8268964>
- [5] Chen S, Chong SR, Kopald H, Levonian Z, Wei Y. Read back error detection using automatic speech recognition. Twelfth USA/Europe Air Traffic Management Research and Development Seminar (ATM2017). <http://doi.org/10.21437/Interspeech.2019-1962>. Available from: http://atmseminar.org/seminarContent/seminar12/papers/12th_ATM_RD_Seminar_paper_20.pdf
- [6] Cerna A, Ehr H, Kleinert M, Helmke H, Kern C, Klakow D, et al. Semi-supervised adaptation of assistant based speech recognition models for different approach areas. 2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC).

- <http://doi.org/10.1109/DASC40568.2018>.
Available from: <https://ieeexplore.ieee.org/document/8569879>
- [7] Lin Y. Spoken instruction understanding in air traffic control: challenge, technique, and application. *Aerospace*. 2021. <http://doi.org/10.3390/aerospace8030065>
Available from: <https://www.mdpi.com/journal/aerospace>.
- [8] Helmke H, Klakow D, Schmidt A, Schulder M, Oualil Y. Real-time integration of dynamic context information for improving automatic speech recognition. *INTERSPEECH 2015*. Available from: https://isca-speech.org/archive/interspeech_2015/papers/i15_2107.pdf
- [9] Holone H, Nguyen V. Possibilities, challenges, and the state of the art of automatic speech recognition in air traffic control. *World Acad Sci Eng Technol Int J Comput Inf Eng*. 2015;9(8). <http://doi.org/10.5281/zenodo.1108428>
- [10] Bakr M. A linguistic approach to marine communication. *J Navig*. 1979;32(2):171-179. <http://doi.org/10.1017/S0373463300037887>
- [11] Narayanan S, Balasundaram SR. Error detection using syntactic analysis for air traffic speech. *Proceedings of the Second International Conference on Computing, Communications, and Cyber-Security. Lecture Notes in Networks and Systems*, vol 203. 2021.
- [12] Delpech E, Lancelot F, Farinas J, Pellegrini T. The Airbus air traffic control speech recognition 2018 challenge: towards ATC automatic transcription and call sign detection. *InterSpeech 2019*. <http://doi.org/10.21437/Interspeech.2019-1962>
- [13] Lamere P, Kwok P, Gouvêa E, Raj B, Singh R, Walker W, Warmuth MK, Wolf P. The CMU SPHINX-4 speech recognition system. 2001.
- [14] Hofbauer K, Petrik S, Hering H. The ATCOSIM corpus of non-prompted clean air traffic control speech. 2008. Available from: <http://www.lrec-conf.org/lrec2008/>
- [15] Helmke H, Himawan I, Motlicek P, Oualil Y, Srinivasamurthy A, Szaszák G. Semi-supervised learning with semantic knowledge extraction for improved speech recognition in air traffic control. *INTERSPEECH 2017*. <http://doi.org/10.21437/Interspeech.2017-1446>