NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Keyword and Named Entity Recognition on Air Traffic Control Text

Andy Ng Chin Kuan

School of Computer Science and Engineering

2019/2020

NANYANG TECHNOLOGICAL UNIVERSITY

CZ4079 Keyword and Named Entity Recognition on Air Traffic Control Text

Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor of Computer Science of Nanyang Technological University

By

Andy Ng Chin Kuan

School of Computer Science and Engineering

2019/2020

Abstract

Airline safety has always been an important aspect of air travel. With recent incidents happening due to miscommunication between pilots and air traffic controllers, it is crucial that we figure out solutions to prevent mishaps from happening in the skies and in the airports. This report will utilize one of the methods of Natural Language Processing (NLP), specifically Named Entity Recognition (NER) on Air Traffic Control (ATC) Conversations. An ATC dataset will be generated using grammar rules using OpenFST and the Thrax Compiler. After that, this dataset will be used to train a model using the Bi-LSTM-CRF based custom named entity system. All named entities will be highlighted and presented on a working application. A speech to text recognizer will also be implemented in the working application to simulate pilots talking to ATC and vice versa.

Acknowledgments

This Final Year Project was made possible due to the support and guidance of several people, who had helped me in terms of their knowledge in Named Entity Recognition, Air Traffic Control Text and Speech Recognition.

Firstly, I would like to express my utmost gratitude to my supervisor, A/P Chng Eng Siong for his great help and advice during the development of the project. Even though I may have made mistakes through the course of the project, he always showed patience and made sure I understand the necessary details of the project. Through this experience, I realized I have been more meticulous in my work.

Secondly, I would also like to thank Mr Andrew Koh Jin Jie, Ms Ho Thi Nga and Ms Hou Nana for their kind support and assistance throughout this project. Whenever I faced any problems, they will not hesitate to help me.

Lastly, I would like to thank Mr Jeremy Thia Ming Xuan for his assistance for the training and generating of grammar and sentences for Air Traffic Control Text. Your valuable help is very much appreciated.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Imagine you are at Changi Airport now. Six hours later, you are in Incheon Airport. That is how convenient and fast air travel is nowadays. Every day, flights are coming in and out of each country and Air Traffic Control (ATC) plays an integral part in preventing collisions of planes in the sky as well as on the airport runways. A small mistake from pilots and ATC will cause disastrous consequences which include the loss of lives and damage to airports. Patty stated that communication failures had caused deaths of more than 2000 people in plane crashes since the mid-1970s [1]. One life lost is one too many and we must ensure that lives are not lost due to miscommunication between pilots and ATC. Patty also added that given radio communication is the primary communication method between ATC and pilots, effective communication is significant for aviation safety [1]. This is where NER on ATC text can help to facilitate communication and prevent potential communication errors. By showing important entities to pilots and controllers in real time, they will be able to detect keywords and spot any discrepancies whenever there is a communication breakdown. This allows both parties to be vigilant to ensure that we have airline safety in the skies and on land. Since there are so many words spoken between the controllers and the pilots, NER helps to solve the problem even more as the recognition of important entities and keywords can be detected in a short period of time.

Name entities that are used normally by pilots and ATC include call sign, runway, flight level and speed of the plane. An example of two highlighted name entities in an ATC sentence is shown below.

Singapore three four two vacate echo five

The yellow highlighted phrase Singapore three four two is the name entity for call sign of the aircraft whereas the green highlighted phrase vacate echo five is the name entity for taxi.

1.2 Objectives and Scope

Performing NER on Air Traffic Control text is a difficult and tedious task since there is no clear-cut solution to obtain perfect named entity recognition on ATC text. However, this project aims to achieve the highest possible accuracy for ATC text so that most ATC texts could be recognized and be useful to pilots and Air Traffic Controllers who could be using this system in the future.

The objectives in this project are as follows:

1. Apply existing natural language processing techniques to detect and annotate named entities from Air Traffic conversations between pilots and Air Traffic Controllers. The focus of the project is the annotation of conversations between pilots and controllers and vice versa. Pilots will communicate with ATC to ensure they get safe landing clearance or request clearance on the runways whereas ATC will approve clearance, use of the runways on the airport and ensure safe landing of airplanes.

2. Train a named entity recognition model which can be used to identify entities when typing examples of ATC text. A demo on the ATC text with the annotated entities will be displayed on a website in real time.

3. Implementation of a speech to text recognizer on the demo website so that when pilots or controllers speak, the system can convert whatever they say into text on the website.

A similar project was conducted by Jeremy Thia Ming Xuan half a year ago. This project is a continuation of his work to improve the generated grammar, increase the number of named entities for both pilots and ATC to be recognized and adding a speech to text recognizer module into the demo website.

1.3 Report organization

This report is organized as follows:

- ❖ Chapter 1 highlights the background, objectives and scope of this project.
- ❖ Chapter 2 shows the research that I have done for this project which includes the definition of natural language processing and the various techniques performed recently to train named entity recognition such as Recurrent Neural Networks (RNN).
- ❖ Chapter 3 focuses on the implementation of the project as well as the design of the ATC text prototype for visualization.
- ❖ Chapter 4 concludes the report where we identify potential problems of the current project and provide solutions to overcome these problems for future use.

Chapter 2

Literature Review

2.1 Natural Language Processing

Natural Language Processing also known as NLP is one of the fields of Artificial Intelligence whereby computers can comprehend, analyze and obtain meaning from human language. Speech and text are how humans communicate with one another. Text varies from web pages, messages, signs and lecture notes. Speech includes various languages that we may use to communicate every day. Through NLP, we can utilize some of its functions such as Part-of-Speech Tagging [2], Named Entity Recognition [3] and Speech Recognition.

Part-of-Speech Tagging refers to the allocation of part-of-speech labels to words in a corpus [2]. The most common POS tagging of words interpreted by humans are Nouns, Adjectives and Verbs. However, some words have different meanings in different context, hence using existing libraries for POS tagging may not be very accurate.

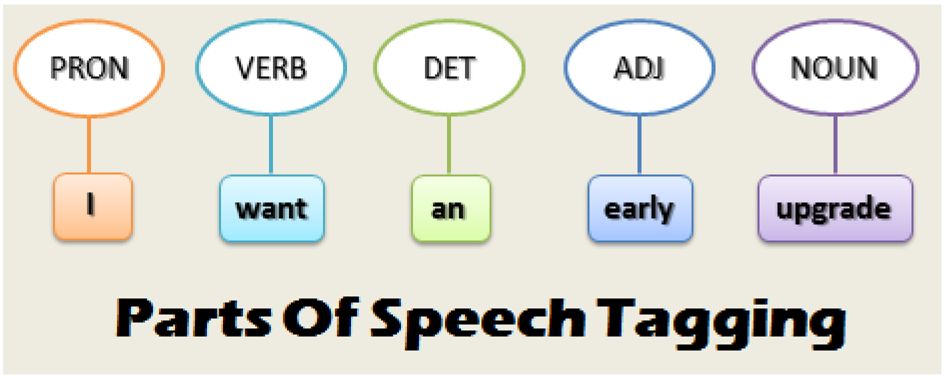
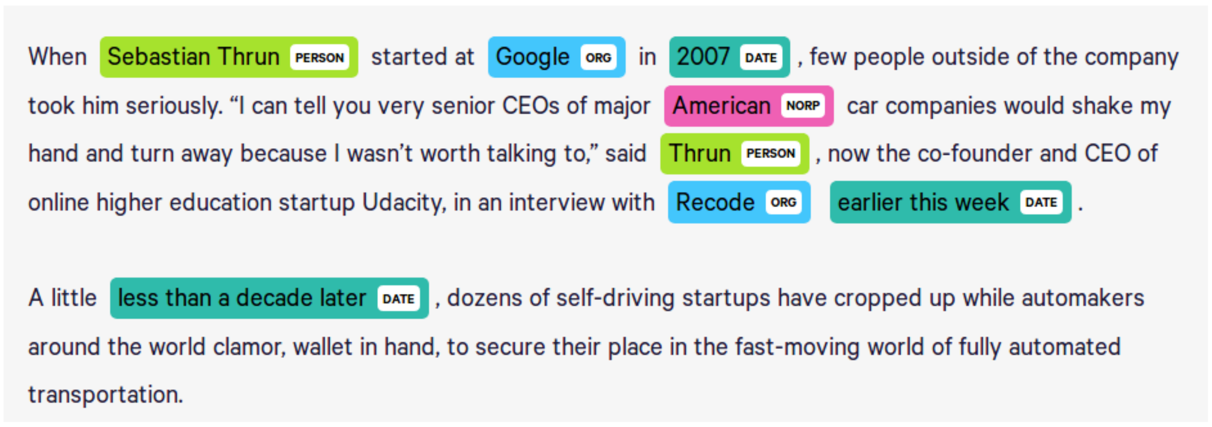


Figure 1: POS tagging of a sentence [4]

Named Entity Recognition (NER) refers to identifying and classifying keywords or named entities located in unstructured text into pre-defined categories such as Person, Location, Organizations, date and monetary values [3]. NER is a sub-task of Information Extraction and is one of the most thoroughly researched in Natural Language Processing.



When Sebastian Thrun PERSON started at Google ORG in 2007 DATE , few people outside of the company took him seriously. "I can tell you very senior CEOs of major American NORP car companies would shake my hand and turn away because I wasn't worth talking to," said Thrun PERSON , now the co-founder and CEO of online higher education startup Udacity, in an interview with Recode ORG earlier this week DATE .

A little less than a decade later DATE , dozens of self-driving startups have cropped up while automakers around the world clamor, wallet in hand, to secure their place in the fast-moving world of fully automated transportation.

Figure 2: Words with NER tags are annotated in different colours in Spacy [5]

Nowadays, NER has been utilized by various industries. NER has been used by human resource departments in companies [6]. Resumes of applicants often have too much information which may not be relevant to what the evaluator wants. With NER, resumes could be evaluated efficiently, simplifying the effort to shortlist candidates. Moreover, news providers have been utilizing NER by helping news providers to scan entire articles and sift out the important details which include the major people, organizations and places [7].

In this project, new entities are created to be implemented in the Air Traffic Control system and these entities include Callsign, Runway, flight level and speed of the aircraft.

Speech recognition is where computers can detect words and phrases in spoken languages such as English, Mandarin and convert them into machine-readable format [8]. Examples of speech recognition include Google Speech and Amazon Transcribe.

2.2 Supervised and Unsupervised Learning

Supervised learning is using an algorithm to learn the mapping function from the input to the output where you have input variables (x) and output variable (Y) [9]. The algorithm makes predictions on the training data continuously and it is validated by a test dataset. The algorithm will eventually achieve an acceptable level of performance and that is where learning will come to a halt.

For unsupervised learning, no labels are given to the learning algorithm [10]. The learning algorithm will be left on its own to generate its own learning. Unsupervised learning can be used to find underlying distributions in the data to learn more about the data [9]. They are normally grouped in clustering and association problems in machine learning.

In this project, supervised learning will be used to train ATC text since we are not solving clustering and association problems. Moreover, ATC text requires annotation of specific words which is used by a learning algorithm to make predictions for future ATC text.

2.3 Recurrent Neural Networks

Recurrent Neural networks (RNN) are models that can process sequential data one element at a time and selectively pass information across sequence steps [11]. They can recall the past and use the past information to influence its decisions.

Vanilla RNN is the basic Recurrent Neural Network which uses a tanh activation function and the RNN cell is referred to as tanh units. The figure below shows the Vanilla RNN before it is unfolded (left) and after it is unfolded (right).

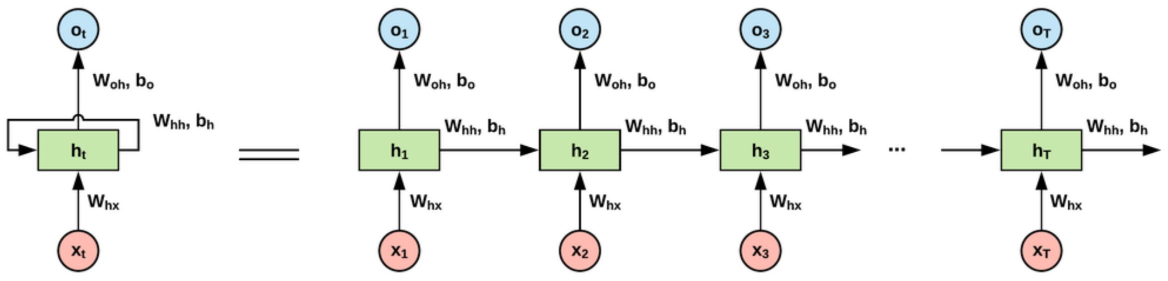


Figure 3: A simple Vanilla RNN

Even though Vanilla RNN could be used as solutions to sequential problems, it is hard to train them to learn long term dynamics and dependencies [12]. This is due to the presence of exploding and vanishing gradients in Vanilla RNN.

2.4 Long Short Term Memory Units (LSTMs)

As mentioned above, due to the presence of exploding and vanishing gradients in Vanilla RNN, the neural networks are not able to learn efficiently. Hence, another variant of RNN is created to counter that problem. This RNN is known as Long Short Term Memory (LSTM) RNN. According to Hochreiter and Schmidhuber, the LSTM architecture consists of memory cells and gate units [13]. These gate units include the input gate, output gate and the forget gate. These gate units help to avoid input weight conflicts which they control error flow to the memory cell.

There is another variant of LSTM which is Bidirectional long short term memory (Bi-LSTM). Bi-LSTM is able to preserve both past and future information whereas the normal LSTM is able to only preserve past information.

2.5 Recent NLP techniques

2.5.1 Word Embeddings

Word Embeddings are numerical representation of words usually in vector shape [14]. The purpose of word embeddings is to categorize words with similar meaning into a same representation. Every individual word is mapped into one vector. After that, these vector values will be learned in a way that is similar to a neural network. According to Mandelbaum, the motivation for word embeddings is that we cannot possibly represent each word with a one-hot vector where one value is unique, and the rest of the values are zero. This will result in a large data sparsity and more data will be required to successfully train statistical models. Hence, through the use word embeddings, we can map semantically similar words to nearby points and make the representation carry the actual word meaning [14].

2.5.2 ELMo

Embeddings from Language Model (ELMo) is type of deep contextualized word representation which models characteristics of word use and how words are used across linguistic contexts [15]. The main disadvantage of word embeddings is that they do not identify polysemy which also refers to words with multiple meanings. One example of polysemy is shown below.

- John wants to borrow a **book**.
- John wants to **book** a flight.

In the first sentence, book refers to a printed work of pages where John wants to borrow. However, in the second sentence, book refers to reserving a flight ticket.

For word embeddings, it only allows one vector for each word which refers to the meaning of the word. Conversely, ELMo identify context in words which the word ‘book’ will result in different vectors mapped. According to Purohit, ELMo utilizes bi-directional LSTM in training which its language model will understand both the next word and the previous word in the sentence [16]. Since ATC text requires linguistic contexts, I will be using the ELMo model instead of word embeddings for training in this project.

2.6 Current Named Entity Recognition systems

As mentioned previously, named entity recognition refers to the labelling and identification of name entities. Before the adoption of neural networks, named entity recognition are done using classical approaches [17]. These classical methods include Hidden Markov Models and Conditional Random Fields (CRF) for Sequence Prediction. One of the more commonly used architecture for named entity recognition systems is Recurrent Neural Network (RNN). More recently, the Bi-LSTM-CNN-CRF model is used for the training of NER systems where it utilizes bidirectional LSTM for its features and decode using the CRF layer with an additional character representation before LSTM that are created using Convolutional Neural Network (CNN) [18]. For instance, a popular Python library, Allennlp uses the Bi-LSTM-CNN-CRF [19] model for Elmo embeddings on NER. According to Peters et al., the Bi-LSTM-CNN-CRF model with ELMo embeddings has a F-score of 92.22 which is better than the previous models implemented by other researches [15]. This goes to show that by using the Bi-LSTM-CNN-CRF model with ELMo embeddings, I am able to get a high accuracy of identifying my named entities for ATC text.

Most of the existing libraries use the CoNLL-2003 [20] dataset for training. The dataset is taken from the Reuters Corpus and has four entity types to be tagged which is Person (PER), Location (LOC), Organization (ORG) and Miscellaneous (MISC). With no pre-trained model for ATC text in popular libraries such as Spacy, Flair and Allennlp since the NER is done on a specific words in the airline industry, we will have to generate our own ATC dataset and use the Bi-LSTM-CNN-CRF model with ELMo representations for training.

2.7 Air Traffic Control (ATC) Text

Air traffic controllers learn a specific set of phraseology and communicate with pilots to ensure safety of passengers on board. I will explain some of the important words and phrases that ATC and pilots use below. These phrases include ICAO alphabets and numbers, Call Sign, Runway, Clearance, Taxi and Frequency.

2.7.1 ICAO alphabets and numbers

ICAO refers to the International Civil Aviation Organization. The ICAO uses radiotelephone spelling alphabet which is a set of words to represent letters of an alphabet in oral communication [21]. This is to prevent mistakes where letters and numbers may sound similar. Such letters include M and N where they can be mistaken for each other. As a result, for example, A will be referred to as Alpha when it is being transmitted via the radio from ATC to pilots and vice versa [22]. The table below will show how every alphabet and number is being pronounced by ATC and pilots.

Alphabet	Radiotelephony
A	Alpha
B	Bravo
C	Charlie
D	Delta
E	Echo
F	Foxtrot
G	Golf
H	Hotel
I	India
J	Juliet
K	Kilo
L	Lima
M	Mike

N	November
O	Oscar
P	Papa
Q	Quebec
R	Romeo
S	Sierra
T	Tango
U	Uniform
V	Victor
W	Whiskey
X	Xray
Y	Yankee
Z	Zulu
0	Zero
1	One
2	Two
3	Three
4	Four
5	Five
6	Six
7	Seven
8	Eight
9	Nine/ Niner

Table 1: Radiotelephony of alphabets and numbers

2.7.2 Call Sign

A call sign is a unique identifier for an airplane. A call sign is normally identified as an airline designator followed by a flight number [23]. For instance, Speedbird one two three refers to British Airways flight number 123 and Singapore four two three refers to Singapore Airlines flight number 423. When ATC communicate on the radio, the presence of these unique call signs makes sure that communication is clear.

2.7.3 Runway

A runway is defined as a strip of land where an aircraft takes off and land. Every runway is named after the compass bearings of the Earth’s magnetic field, which is between 01 to 36. The last number of the compass bearing is removed to get the two numbers for the runway. For example, if the compass bearing is 158 degrees, the number is rounded to the nearest 10 degrees and the last number will be removed [24]. Hence, the runway number will be 16. The number on the opposite side of the runway will be 34 since the direction is 180 degrees from the other. For big airports, they have parallel runways with same numbers. To differentiate these runways, there will be a left, center and a right runway. Therefore, the left, center and right runway will be named 28L, 28C and 28R respectively if the compass bearing is 280 degrees.

2.7.4 Taxi

Taxiing is when an aircraft uses its own engine power to move on the ground. Taxiing occurs when an aircraft has just landed on the airport runway or when it is preparing for takeoff. One such example of ATC communicating with pilots is Singapore One Two Three Taxi Via Runway Three Five Right.

2.7.5 Clearance

According to Mjåtveit, ATC issues clearances to the crew aboard for departure clearance, taxi clearance, takeoff clearance, approach clearance and landing clearance [25]. Whenever ATC gives an instruction to the aircraft by mentioning its callsign and the instructions, the pilots will repeat the instructions followed by their callsign. For departure clearance, before leaving the gate where the aircraft is parked, the flights are given a clearance. A ground controller issues clearance of movement on taxiways on the airport for taxi clearance. A takeoff clearance is needed when the aircraft is allowed to taxi on the runway. An approach clearance is necessary because it is hard to envisage the landing conditions of the aircraft. Therefore, the approach clearance will be given before the landing and taxi clearance at the destination airport [25]. A landing clearance is instructed by the tower controller to the pilots onboard.

2.7.6 Frequency

Every airport will have various radio frequencies that the pilots and ATC will communicate with. These frequencies include ground frequency, departure frequency, approach frequency and tower frequency. All ATC communication frequencies are currently between 118.0MHz and 137.0Mhz [26]. Whenever an aircraft leaves one of the coverage areas, the pilots will have to switch to another radio frequency to communicate with that particular ATC center. For example, if an aircraft wants to cross the runway, the pilot has to switch from the ground frequency to the tower frequency because active runways are under the control of the tower controller.

Chapter 3

System Implementation and Design

This chapter provides an overview of the implemented system for the named entity recognition of Air Traffic Control Text. This includes the generation of a dataset and using this dataset to train a model. After that, a demo website will show the result of the model trained.

3.1 System Architecture

The system architecture is a client and server model. The client will interact with the natural language processing module which includes the name entity recognizer and the speech to text recognizer in the server. Whenever an input text is provided by the client, the name entity recognizer returns the input text with the annotated name entities. On the other hand, whenever the speech to text function is selected in the client side, the server will return the generated speech to text and the annotated name entities to the client. The named entity highlighter will highlight the named entities which is returned to the client. The figure below will show the system architecture that will be implemented in this project.

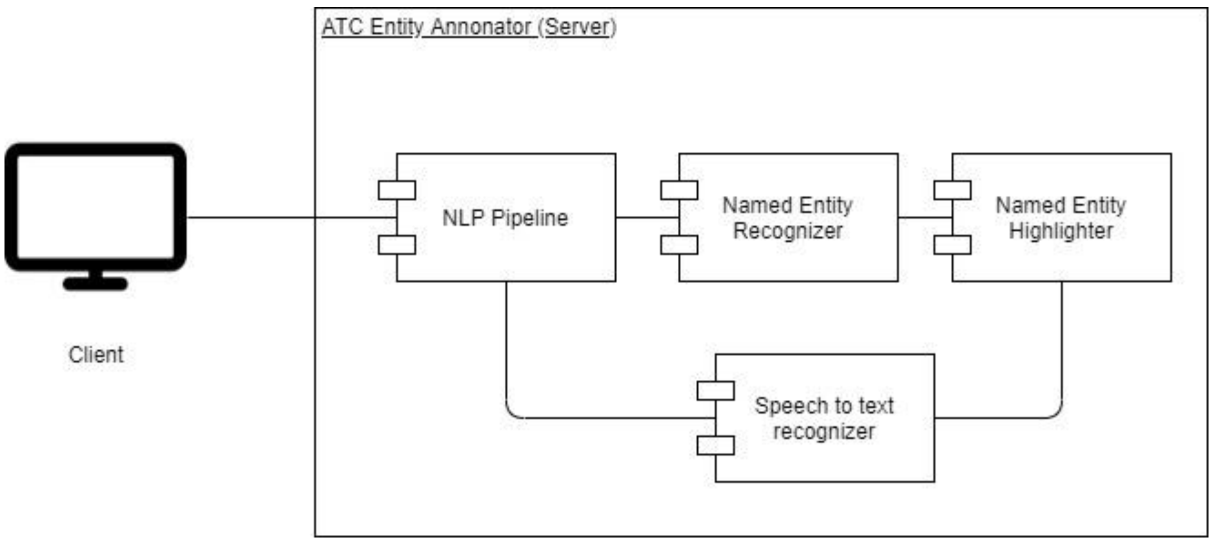
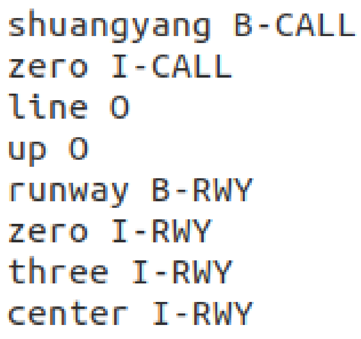


Figure 4: System architecture of ATC Entity Annotator

3.2 Name entity recognizer

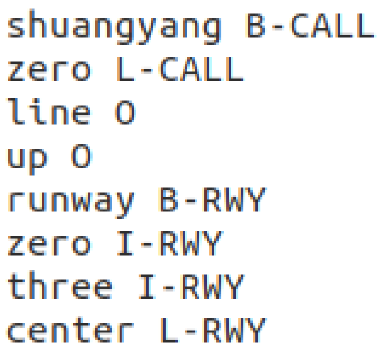
In order to perform name entity recognition on ATC text, we will need to be able to tag the words in the dataset. A common tagging scheme, BIO (Beginning – Inside – Outside) is normally used for name entity recognition. Beginning stands for the start of the name entity that we want to tag; Inside refers to the middle or the end of the name entity we want to tag and Outside refers to the words that we do not want to tag [27]. The figure below will show a simple example on the BIO scheme.



```
shuangyang B-CALL
zero I-CALL
line O
up O
runway B-RWY
zero I-RWY
three I-RWY
center I-RWY
```

Figure 5: BIO tagging of ATC text

In this project, a more detailed scheme than BIO will be used, the BILOU (Beginning – Inside – Last – Outside – Unit) scheme. BILOU is used since it provides a better performance for name entity recognition. The difference between BIO and BILOU is that the last token that is tagged as a name entity will be tagged as Last and if the name entity is only one token, then it will be labeled as Unit. The figure below will show an example of the BILOU scheme that will be used for ATC text.



```
shuangyang B-CALL
zero L-CALL
line O
up O
runway B-RWY
zero I-RWY
three I-RWY
center L-RWY
```

Figure 6: BILOU tagging of ATC text

The model used in this project is a Bi-LSTM-CNN with Conditional Random Field (CRF) decoder. This model was introduced by Peters et. al. [15] in the paper which mentioned about ELMo and was used by Jeremy [28].

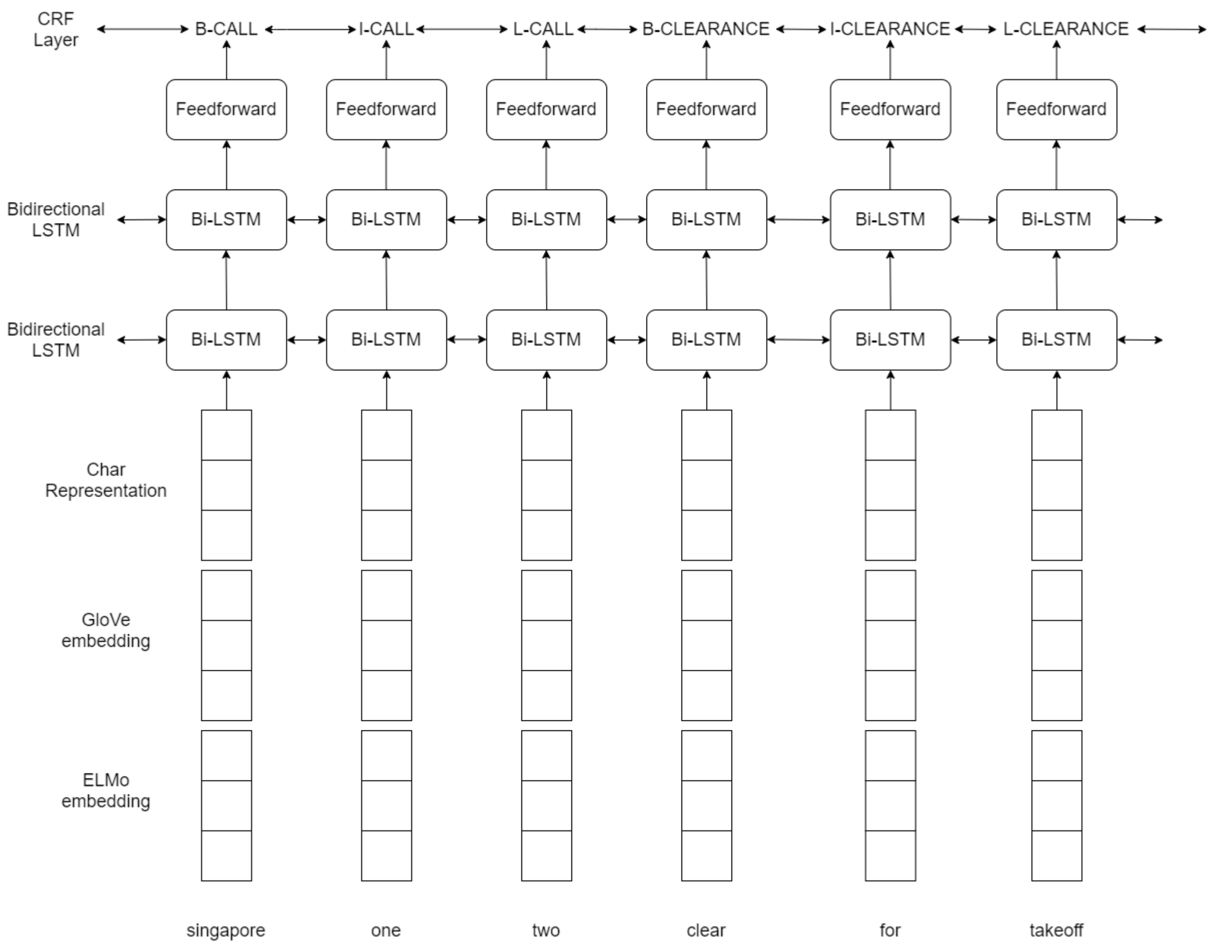


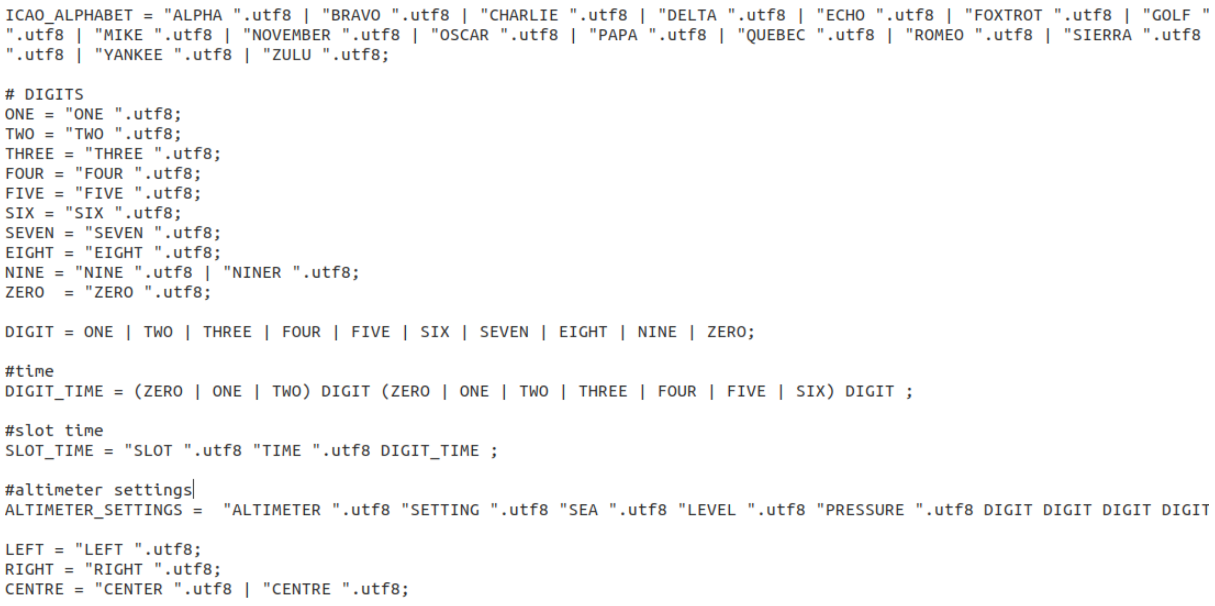
Figure 7: Bi-LSTM-CRF decoder

Figure 7 represents a quick glimpse of the ELMo model that is used to train the ATC text. The ATC text is located at the bottom of the figure. The words are embedded into the vector space using the GloVe [29] embeddings, ELMo embeddings and its character representation. After that, the vector features were added into the 2-layer bidirectional LSTM. The output of the second layer of bidirectional LSTM will be given to the feedforward neural network. Lastly, the CRF decoder will use the output of the feedforward neural network for decoding.

3.3 Generation of dataset

For annotation of ATC text, we will need a large dataset. There are two ways to obtain the annotated ATC text. Firstly, we can find actual Air Traffic transcripts and annotate them manually or through a script. However, this is not a viable option since some of these transcripts are confidential and need money to purchase. Secondly, we can generate Air Traffic transcripts through a context free grammar to simulate conversations between the pilots and ATC. After generating the dataset, we can use it to compare with real ATC text on live ATC radio or ATC manuals. If there are any errors, we can perform corrections in the context free grammar to generate the correct dataset.

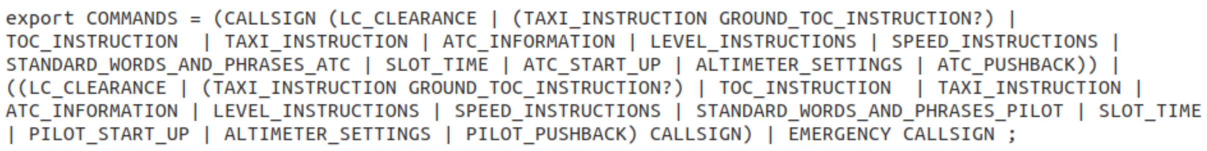
OpenFST [30] provides us a way to generate the ATC dataset. It comprises of a Thrax Grammar Compiler. This compiler will compile regular expressions and context-dependent rewrite rules into FST archives (fars) of weighted finite-state transducers [31]. String FSTs are enclosed by double quotes in Thrax. We can also specify weights in the grammar rules using the notation "<>" so that a higher weightage specified will allow those tokens to appear more frequently in the generated dataset. An example of a weighted expression is Singapore<0.005>. This expression will add weight 0.005 to Singapore. The figure below will show some of the grammar rules that are written and saved in a grammar file.



```
ICAO_ALPHABET = "ALPHA ".utf8 | "BRAVO ".utf8 | "CHARLIE ".utf8 | "DELTA ".utf8 | "ECHO ".utf8 | "FOXTROT ".utf8 | "GOLF "
".utf8 | "MIKE ".utf8 | "NOVEMBER ".utf8 | "OSCAR ".utf8 | "PAPA ".utf8 | "QUEBEC ".utf8 | "ROMEO ".utf8 | "SIERRA ".utf8
".utf8 | "YANKEE ".utf8 | "ZULU ".utf8;

# DIGITS
ONE = "ONE ".utf8;
TWO = "TWO ".utf8;
THREE = "THREE ".utf8;
FOUR = "FOUR ".utf8;
FIVE = "FIVE ".utf8;
SIX = "SIX ".utf8;
SEVEN = "SEVEN ".utf8;
EIGHT = "EIGHT ".utf8;
NINE = "NINE ".utf8 | "NINER ".utf8;
ZERO  = "ZERO ".utf8;

DIGIT = ONE | TWO | THREE | FOUR | FIVE | SIX | SEVEN | EIGHT | NINE | ZERO;

#time
DIGIT_TIME = (ZERO | ONE | TWO) DIGIT (ZERO | ONE | TWO | THREE | FOUR | FIVE | SIX) DIGIT ;

#slot time
SLOT_TIME = "SLOT ".utf8 "TIME ".utf8 DIGIT_TIME ;

#altimeter settings
ALTIMETER_SETTINGS =  "ALTIMETER ".utf8 "SETTING ".utf8 "SEA ".utf8 "LEVEL ".utf8 "PRESSURE ".utf8 DIGIT DIGIT DIGIT DIGIT

LEFT = "LEFT ".utf8;
RIGHT = "RIGHT ".utf8;
CENTRE = "CENTER ".utf8 | "CENTRE ".utf8;
```

Figure 8: Some grammar rules in grammar file

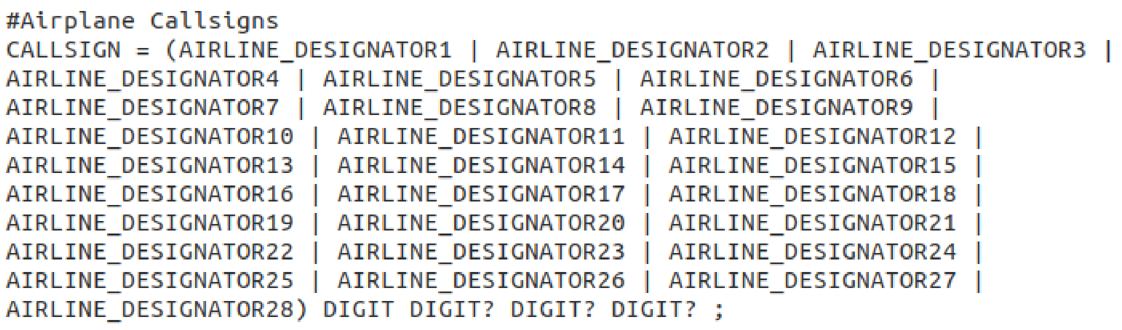
Based on Figure 8, the non-terminal symbol is located at the left of the equal sign whereas the tokens are located at the right of the equal sign. All the grammar rules will be exported at the last line of the grammar file. These grammar rules will be used to generate the sentences that will be used for annotation. The figure below will show how the grammar rules are being exported.



```
export COMMANDS = (CALLSIGN (LC_CLEARANCE | (TAXI_INSTRUCTION GROUND_TOC_INSTRUCTION?) |
TOC_INSTRUCTION  | TAXI_INSTRUCTION | ATC_INFORMATION | LEVEL_INSTRUCTIONS | SPEED_INSTRUCTIONS |
STANDARD_WORDS_AND_PHRASES_ATC | SLOT_TIME | ATC_START_UP | ALTIMETER_SETTINGS | ATC_PUSHBACK)) |
((LC_CLEARANCE | (TAXI_INSTRUCTION GROUND_TOC_INSTRUCTION?) | TOC_INSTRUCTION  | TAXI_INSTRUCTION |
ATC_INFORMATION | LEVEL_INSTRUCTIONS | SPEED_INSTRUCTIONS | STANDARD_WORDS_AND_PHRASES_PILOT | SLOT_TIME
| PILOT_START_UP | ALTIMETER_SETTINGS | PILOT_PUSHBACK) CALLSIGN) | EMERGENCY CALLSIGN ;
```

Figure 9: Exported grammar rules

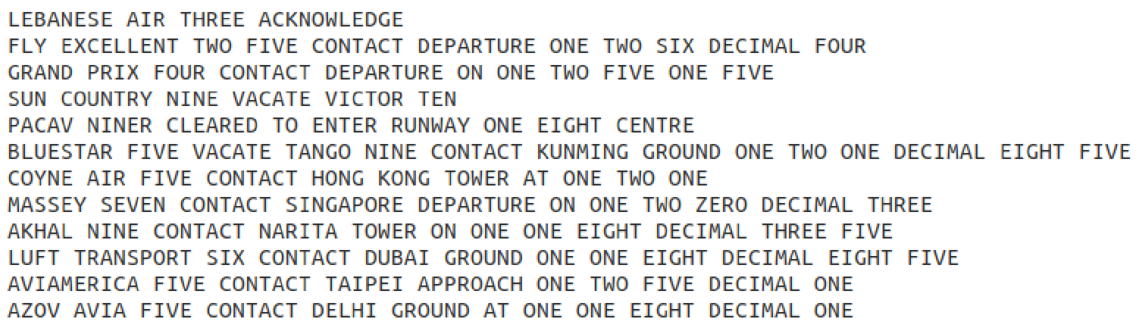
For the grammar rules of callsign, a Python script was developed to add any possible airline designators into the grammar file. Since there are more than 5000 call signs, we cannot possibly type every single airline designator manually in the grammar file. This Python script will read all the airline designators that are placed in a csv file and convert into the format that is acceptable in the grammar file. The total number of airline designators will be split into multiples of 200 to prevent deep recursion overflow in the grammar file. Figure 10 shows how the Callsign grammar rule is being implemented.



```
#Airplane Callsigns
CALLSIGN = (AIRLINE_DESIGNATOR1 | AIRLINE_DESIGNATOR2 | AIRLINE_DESIGNATOR3 |
AIRLINE_DESIGNATOR4 | AIRLINE_DESIGNATOR5 | AIRLINE_DESIGNATOR6 |
AIRLINE_DESIGNATOR7 | AIRLINE_DESIGNATOR8 | AIRLINE_DESIGNATOR9 |
AIRLINE_DESIGNATOR10 | AIRLINE_DESIGNATOR11 | AIRLINE_DESIGNATOR12 |
AIRLINE_DESIGNATOR13 | AIRLINE_DESIGNATOR14 | AIRLINE_DESIGNATOR15 |
AIRLINE_DESIGNATOR16 | AIRLINE_DESIGNATOR17 | AIRLINE_DESIGNATOR18 |
AIRLINE_DESIGNATOR19 | AIRLINE_DESIGNATOR20 | AIRLINE_DESIGNATOR21 |
AIRLINE_DESIGNATOR22 | AIRLINE_DESIGNATOR23 | AIRLINE_DESIGNATOR24 |
AIRLINE_DESIGNATOR25 | AIRLINE_DESIGNATOR26 | AIRLINE_DESIGNATOR27 |
AIRLINE_DESIGNATOR28) DIGIT DIGIT? DIGIT? DIGIT? ;
```

Figure 10: Grammar rule for Callsign

After completing the grammar file, the Thrax Grammar Compiler is used to compile the grammar file. Figure 11 will show some of the sentences that the grammar file has generated.



```
LEBANESE AIR THREE ACKNOWLEDGE
FLY EXCELLENT TWO FIVE CONTACT DEPARTURE ONE TWO SIX DECIMAL FOUR
GRAND PRIX FOUR CONTACT DEPARTURE ON ONE TWO FIVE ONE FIVE
SUN COUNTRY NINE VACATE VICTOR TEN
PACAV NINER CLEARED TO ENTER RUNWAY ONE EIGHT CENTRE
BLUESTAR FIVE VACATE TANGO NINE CONTACT KUNMING GROUND ONE TWO ONE DECIMAL EIGHT FIVE
COYNE AIR FIVE CONTACT HONG KONG TOWER AT ONE TWO ONE
MASSEY SEVEN CONTACT SINGAPORE DEPARTURE ON ONE TWO ZERO DECIMAL THREE
AKHAL NINE CONTACT NARITA TOWER ON ONE ONE EIGHT DECIMAL THREE FIVE
LUFT TRANSPORT SIX CONTACT DUBAI GROUND ONE ONE EIGHT DECIMAL EIGHT FIVE
AVIAMERICA FIVE CONTACT TAIPEI APPROACH ONE TWO FIVE DECIMAL ONE
AZOV AVIA FIVE CONTACT DELHI GROUND AT ONE ONE EIGHT DECIMAL ONE
```

Figure 11: Sentences generated by the grammar file

The sentences will then be annotated using the BILOU scheme. Once all the sentences are annotated, it will be used for training a model.

Below are the entities to be labelled:

Callsign – e.g. speedbird two three one

Slot Time – e.g. slot time zero niner five six

Altimeter – e.g. altimeter setting sea level pressure five eight four one

Runway – e.g. runway one two left

Wind Speed – e.g. wind two zero five knots

Tower – e.g. contact Singapore departure at one two zero decimal three

Emergency – e.g. mayday mayday mayday

Flight Level – e.g. climb and maintain six thousand feet

Speed – e.g. increase speed to nine hundred knots

Taxi – e.g. taxi to holding point

Clearance – e.g. clear for takeoff

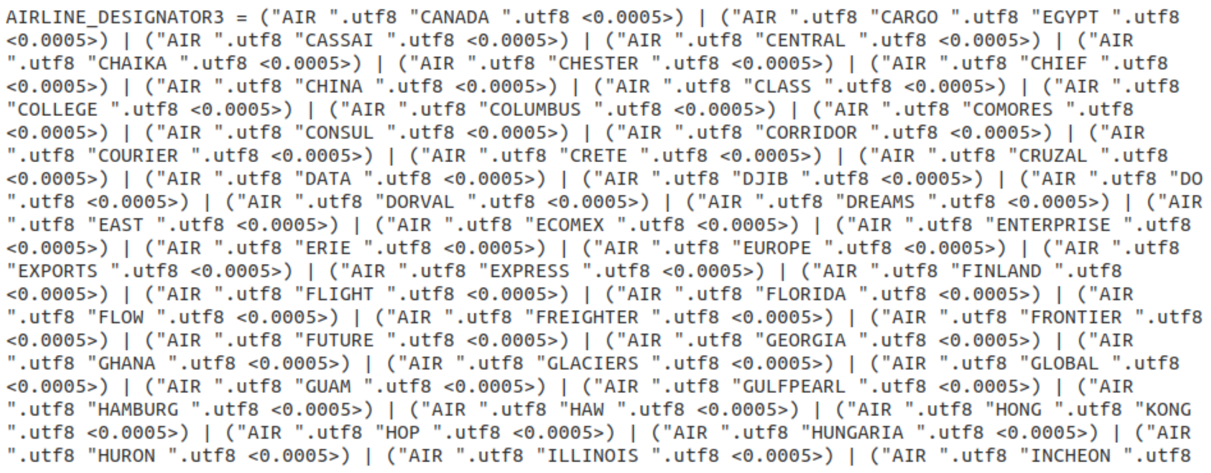
Startup – e.g. ready to start up

Pushback – e.g. request pushback

The number of Airline Designators in the Callsign grammar rule has been increased to more than 5000. For the Runway grammar rule, the number of runways has been increased to recognize all possible runways. The new entities which are added into the grammar file are Slot Time, Altimeter, Emergency, Flight Level, Speed, Startup and Pushback. Previously, the grammar file contains only speech where ATC speaks to pilots only but not vice versa. Now, the grammar file has been changed to cater to pilots speaking back to ATC.

Below are the screenshots for the updated and the new grammar rules that are implemented in the grammar file.

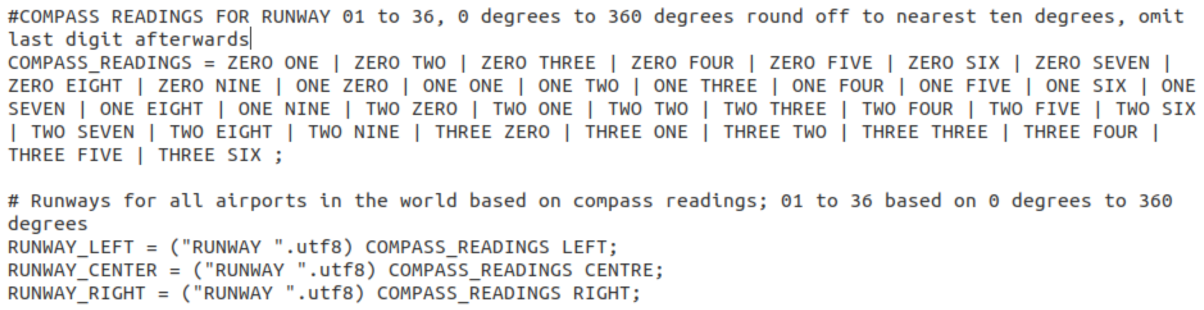
Figure 12 shows some of the airline designators specified in the grammar file. As the number of airline designators is too large, a screenshot of several airline designators will be shown. A Callsign will be one of these airline designators followed by a number.



```
AIRLINE_DESIGNATOR3 = ("AIR ".utf8 "CANADA ".utf8 <0.0005>) | ("AIR ".utf8 "CARGO ".utf8 "EGYPT ".utf8
<0.0005>) | ("AIR ".utf8 "CASSAI ".utf8 <0.0005>) | ("AIR ".utf8 "CENTRAL ".utf8 <0.0005>) | ("AIR
".utf8 "CHAIKA ".utf8 <0.0005>) | ("AIR ".utf8 "CHESTER ".utf8 <0.0005>) | ("AIR ".utf8 "CHIEF ".utf8
<0.0005>) | ("AIR ".utf8 "CHINA ".utf8 <0.0005>) | ("AIR ".utf8 "CLASS ".utf8 <0.0005>) | ("AIR ".utf8
"COLLEGE ".utf8 <0.0005>) | ("AIR ".utf8 "COLUMBUS ".utf8 <0.0005>) | ("AIR ".utf8 "COMORES ".utf8
<0.0005>) | ("AIR ".utf8 "CONSUL ".utf8 <0.0005>) | ("AIR ".utf8 "CORRIDOR ".utf8 <0.0005>) | ("AIR
".utf8 "COURIER ".utf8 <0.0005>) | ("AIR ".utf8 "CRETE ".utf8 <0.0005>) | ("AIR ".utf8 "CRUZAL ".utf8
<0.0005>) | ("AIR ".utf8 "DATA ".utf8 <0.0005>) | ("AIR ".utf8 "DJIB ".utf8 <0.0005>) | ("AIR ".utf8 "DO
".utf8 <0.0005>) | ("AIR ".utf8 "DORVAL ".utf8 <0.0005>) | ("AIR ".utf8 "DREAMS ".utf8 <0.0005>) | ("AIR
".utf8 "EAST ".utf8 <0.0005>) | ("AIR ".utf8 "ECOMEX ".utf8 <0.0005>) | ("AIR ".utf8 "ENTERPRISE ".utf8
<0.0005>) | ("AIR ".utf8 "ERIE ".utf8 <0.0005>) | ("AIR ".utf8 "EUROPE ".utf8 <0.0005>) | ("AIR ".utf8
"EXPORTS ".utf8 <0.0005>) | ("AIR ".utf8 "EXPRESS ".utf8 <0.0005>) | ("AIR ".utf8 "FINLAND ".utf8
<0.0005>) | ("AIR ".utf8 "FLIGHT ".utf8 <0.0005>) | ("AIR ".utf8 "FLORIDA ".utf8 <0.0005>) | ("AIR
".utf8 "FLOW ".utf8 <0.0005>) | ("AIR ".utf8 "FREIGHTER ".utf8 <0.0005>) | ("AIR ".utf8 "FRONTIER ".utf8
<0.0005>) | ("AIR ".utf8 "FUTURE ".utf8 <0.0005>) | ("AIR ".utf8 "GEORGIA ".utf8 <0.0005>) | ("AIR
".utf8 "GHANA ".utf8 <0.0005>) | ("AIR ".utf8 "GLACIERS ".utf8 <0.0005>) | ("AIR ".utf8 "GLOBAL ".utf8
<0.0005>) | ("AIR ".utf8 "GUAM ".utf8 <0.0005>) | ("AIR ".utf8 "GULFPEARL ".utf8 <0.0005>) | ("AIR
".utf8 "HAMBURG ".utf8 <0.0005>) | ("AIR ".utf8 "HAW ".utf8 <0.0005>) | ("AIR ".utf8 "HONG ".utf8 "KONG
".utf8 <0.0005>) | ("AIR ".utf8 "HOP ".utf8 <0.0005>) | ("AIR ".utf8 "HUNGARIA ".utf8 <0.0005>) | ("AIR
".utf8 "HURON ".utf8 <0.0005>) | ("AIR ".utf8 "ILLINOIS ".utf8 <0.0005>) | ("AIR ".utf8 "INCHEON ".utf8
```

Figure 12: Grammar for some airline designators

As mentioned above in the Literature Review for ATC text, the runway name will be runway followed by a compass reading from 01 to 36.



```
#COMPASS READINGS FOR RUNWAY 01 to 36, 0 degrees to 360 degrees round off to nearest ten degrees, omit
last digit afterwards
COMPASS_READINGS = ZERO ONE | ZERO TWO | ZERO THREE | ZERO FOUR | ZERO FIVE | ZERO SIX | ZERO SEVEN |
ZERO EIGHT | ZERO NINE | ONE ZERO | ONE ONE | ONE TWO | ONE THREE | ONE FOUR | ONE FIVE | ONE SIX | ONE
SEVEN | ONE EIGHT | ONE NINE | TWO ZERO | TWO ONE | TWO TWO | TWO THREE | TWO FOUR | TWO FIVE | TWO SIX
| TWO SEVEN | TWO EIGHT | TWO NINE | THREE ZERO | THREE ONE | THREE TWO | THREE THREE | THREE FOUR |
THREE FIVE | THREE SIX ;

# Runways for all airports in the world based on compass readings; 01 to 36 based on 0 degrees to 360
degrees
RUNWAY_LEFT = ("RUNWAY ".utf8) COMPASS_READINGS LEFT;
RUNWAY_CENTER = ("RUNWAY ".utf8) COMPASS_READINGS CENTRE;
RUNWAY_RIGHT = ("RUNWAY ".utf8) COMPASS_READINGS RIGHT;
```

Figure 13: Grammar for runway

Slot time is issued by the ATC when the airspace is limited by the amount of traffic.

```
#slot time
SLOT_TIME = "SLOT ".utf8 "TIME ".utf8 DIGIT_TIME ;
```

Figure 14: Grammar for Slot Time

Altimeter of an aircraft tells pilots how high they are flying and what level is required for them for a safe flight.

```
#altimeter settings
ALTIMETER_SETTINGS =  "ALTIMETER ".utf8 "SETTING ".utf8 "SEA ".utf8 "LEVEL ".utf8 "PRESSURE ".utf8 DIGIT DIGIT DIGIT DIGIT ;
```

Figure 15: Grammar for Altimeter

Emergency instructions normally starts with mayday mayday mayday or pan-pan, pan-pan, pan-pan.

```
#Emergency instructions
MAYDAY = "MAYDAY ".utf8 "MAYDAY ".utf8 "MAYDAY ".utf8 ;
PAN_PAN = "PAN-PAN ".utf8 "PAN-PAN ".utf8 "PAN-PAN ".utf8 ;
EMERGENCY = (MAYDAY | PAN_PAN) ;
```

Figure 16: Grammar for Emergency

Flight level is the measure of the aircraft's altitude. Normally, instructions are climb, ascend, descend and maintain flight level which is shown by the grammar rules in the figure below.

```
#FLIGHT LEVEL
FL_LEVEL = ("FLIGHT ".utf8 "LEVEL ".utf8 ((DIGIT? ("HUNDRED ".utf8 | "THOUSAND ".utf8)) | (DIGIT?
DIGIT?) | (DIGIT? DIGIT? "THOUSAND ".utf8) | (DIGIT? "THOUSAND ".utf8 DIGIT? "HUNDRED ".utf8))) | (DIGIT
("HUNDRED ".utf8 | "THOUSAND ".utf8) "FEET ".utf8) ;

#Level instructions
LEVEL_INSTRUCTIONS = ("CLIMB ".utf8 | "CLIMBING ".utf8 | "ASCEND ".utf8 | "ASCENDING ".utf8 | "DESCEND
".utf8 | "DESCENDING ".utf8) ("TO ".utf8 FL_LEVEL | "AND ".utf8 "MAINTAIN ".utf8 FL_LEVEL) ;
```

Figure 17: Grammar for Flight Level

Speed of aircraft is always some number followed by knots. Normally, instructions are maintaining present speed, decrease or increase speed.

```
#Speed instructions
SPEED = (DIGIT DIGIT DIGIT | DIGIT "HUNDRED ".utf8) ("KNOTS ".utf8 | "KT ".utf8) ;
SPEED_INSTRUCTIONS = ("MAINTAIN ".utf8 "PRESENT ".utf8 "SPEED ".utf8 | (("DECREASE ".utf8 "SPEED ".utf8
"TO ".utf8 | "INCREASE ".utf8 "SPEED ".utf8 "TO ".utf8) SPEED)) ;
```

Figure 18: Grammar for Speed

Whenever the pilots want to start the engines of the aircraft, they have to seek approval from the ATC. The figure below shows how a pilot and ATC communicate when the pilot starts up the aircraft engine.

```
#Pilot request start plane engines
PILOT_START_UP = "READY ".utf8 "TO ".utf8 "START ".utf8 "UP ".utf8 ;

#ATC approves start up
ATC_START_UP = "START ".utf8 "UP ".utf8 "APPROVED ".utf8 ;
```

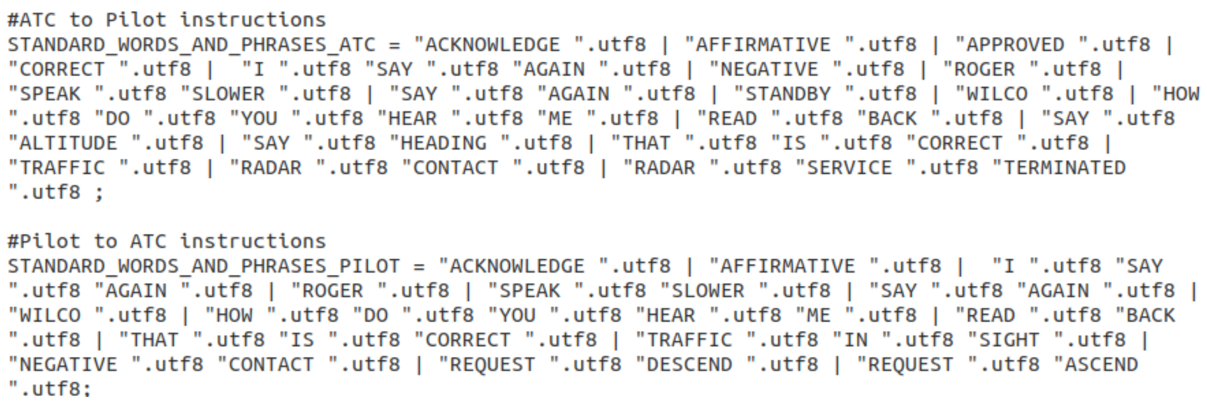
Figure 19: Grammar for startup

Figure 20 shows the grammar for pushback of the aircraft. Before departure, the aircraft must be pushed backwards by tugs. Both the pilot's and ATC communication methods are listed in the grammar below.

```
#Push-back, aircraft have to be pushed backwards by tugs before they can taxi for departure

PILOT_REQUEST_PUSHBACK = "REQUEST ".utf8 "PUSHBACK ".utf8 ;
ATC_APPROVE_PUSHBACK = "PUSHBACK ".utf8 "APPROVED ".utf8 ;
ATC_STANDBY_PUSHBACK = "STAND ".utf8 "BY ".utf8 "EXPECT ".utf8 DIGIT "MINUTES ".utf8 "DELAY ".utf8 ;
PILOT_READY_FOR_PUSHBACK = "READY ".utf8 "FOR ".utf8 "PUSHBACK ".utf8 ;
ATC_BRAKES_RELEASED = "CONFIRM ".utf8 "BRAKES ".utf8 "RELEASED ".utf8 ;
PILOT_BRAKES_RELEASED = "BRAKES ".utf8 "RELEASED ".utf8 ;
ATC_PUSH_BACK_COMPLETED = "PUSHBACK ".utf8 "COMPLETED ".utf8 "CONFIRM ".utf8 "BRAKES ".utf8 "SET ".utf8 ;

PILOT_PUSHBACK = (PILOT_REQUEST_PUSHBACK | PILOT_READY_FOR_PUSHBACK | PILOT_BRAKES_RELEASED) ;
ATC_PUSHBACK = (ATC_APPROVE_PUSHBACK | ATC_STANDBY_PUSHBACK | ATC_BRAKES_RELEASED | ATC_PUSH_BACK_COMPLETED) ;
```

Figure 20: Grammar for Pushback

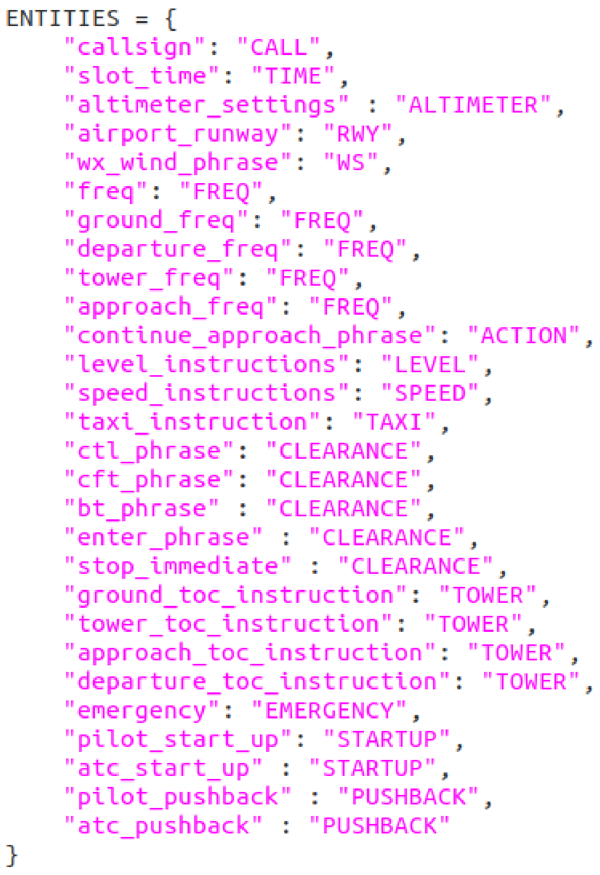
Some words are added and not labelled to get an even distribution of both named entities and non-named entities in the dataset. One such example is when ATC request pilots to speak slower. Speak slower will not be annotated as a name entity. Figure 21 will show the words that will not be annotated as a named entity.



```
#ATC to Pilot instructions
STANDARD_WORDS_AND_PHRASES_ATC = "ACKNOWLEDGE ".utf8 | "AFFIRMATIVE ".utf8 | "APPROVED ".utf8 |
"CORRECT ".utf8 |  "I ".utf8 "SAY ".utf8 "AGAIN ".utf8 | "NEGATIVE ".utf8 | "ROGER ".utf8 |
"SPEAK ".utf8 "SLOWER ".utf8 | "SAY ".utf8 "AGAIN ".utf8 | "STANDBY ".utf8 | "WILCO ".utf8 | "HOW
".utf8 "DO ".utf8 "YOU ".utf8 "HEAR ".utf8 "ME ".utf8 | "READ ".utf8 "BACK ".utf8 | "SAY ".utf8
"ALTITUDE ".utf8 | "SAY ".utf8 "HEADING ".utf8 | "THAT ".utf8 "IS ".utf8 "CORRECT ".utf8 |
"TRAFFIC ".utf8 | "RADAR ".utf8 "CONTACT ".utf8 | "RADAR ".utf8 "SERVICE ".utf8 "TERMINATED
".utf8 ;

#Pilot to ATC instructions
STANDARD_WORDS_AND_PHRASES_PILOT = "ACKNOWLEDGE ".utf8 | "AFFIRMATIVE ".utf8 |  "I ".utf8 "SAY
".utf8 "AGAIN ".utf8 | "ROGER ".utf8 | "SPEAK ".utf8 "SLOWER ".utf8 | "SAY ".utf8 "AGAIN ".utf8 |
"WILCO ".utf8 | "HOW ".utf8 "DO ".utf8 "YOU ".utf8 "HEAR ".utf8 "ME ".utf8 | "READ ".utf8 "BACK
".utf8 | "THAT ".utf8 "IS ".utf8 "CORRECT ".utf8 | "TRAFFIC ".utf8 "IN ".utf8 "SIGHT ".utf8 |
"NEGATIVE ".utf8 "CONTACT ".utf8 | "REQUEST ".utf8 "DESCEND ".utf8 | "REQUEST ".utf8 "ASCEND
".utf8;
```

Figure 21: Words that will be not annotated as a name entity

When the grammar has been finalized, we can generate a new dataset which will be used for training. We will need to annotate the entities for ATC for recognition and tagging. The figure below will show the various grammar rules in the grammar file that will be annotated with its named entity.



```
ENTITIES = {
    "callsign": "CALL",
    "slot_time": "TIME",
    "altimeter_settings" : "ALTIMETER",
    "airport_runway": "RWY",
    "wx_wind_phrase": "WS",
    "freq": "FREQ",
    "ground_freq": "FREQ",
    "departure_freq": "FREQ",
    "tower_freq": "FREQ",
    "approach_freq": "FREQ",
    "continue_approach_phrase": "ACTION",
    "level_instructions": "LEVEL",
    "speed_instructions": "SPEED",
    "taxi_instruction": "TAXI",
    "ctl_phrase": "CLEARANCE",
    "cft_phrase": "CLEARANCE",
    "bt_phrase" : "CLEARANCE",
    "enter_phrase" : "CLEARANCE",
    "stop_immediate" : "CLEARANCE",
    "ground_toc_instruction": "TOWER",
    "tower_toc_instruction": "TOWER",
    "approach_toc_instruction": "TOWER",
    "departure_toc_instruction": "TOWER",
    "emergency": "EMERGENCY",
    "pilot_start_up": "STARTUP",
    "atc_start_up" : "STARTUP",
    "pilot_pushback" : "PUSHBACK",
    "atc_pushback" : "PUSHBACK"
}
```

Figure 22: ATC named entities for recognition

A total of 1 million ATC sentences were annotated. 80% of the sentences were used for training, 10% for the validation dataset and the last 10% for the test dataset.

3.4 Model trained and Evaluation

The model was trained using Pytorch [32] and Allennlp [33] library. Allennlp contains a configuration file for name entity recognition. This configuration file uses ELMo to train a model. The model was trained using Google Colab with the GPU function, was set up to train for 5 epochs and configured to stop early. The table below will show what Jeremy has trained in the previous model:

Previously trained model details:	
Training set size	9,800,000
Validation set size	100,000
Epochs	5
Batch Size	64
L2 weight decay	alpha: 0.4
Optimizer	Adam
Learning rate	10^{-3}
Bi-LSTM Hidden state size	200
Bi-LSTM Inter-layer dropout probability	0.5
Bi-LSTM Number of layers	2
Input dropout probability	0.5
Output dropout probability	0.5
Character representation convolution Number of filter sizes	1
Character representation convolution Filter sizes	{1}
Character representation convolution number of filters	128

ELMO pre-trained embedding dimension	1024
GloVe pre-trained embedding dimension	50

Table 2: Details for previously trained model

The model previously trained by Jeremy can only detect ATC communicating to pilots. However, if the call sign is located at the end of the sentence, the model will not be able to detect it as a name entity. Moreover, there is a limited amount of airline designators in the grammar file. When the training dataset is huge, in this case, 9.8 million sentences, overfitting will occur. This means that the model will only annotate specific call signs found in the dataset. On the other hand, if we train a model with a dataset of 100,000 sentences, underfitting will occur and some of the call signs will not be detected since the model has not seen it frequently. Therefore, to get a more generalized model, 800,000 sentences will be trained, and 100,000 sentences will be used for validation. The final trained model details are shown below.

Final Trained Model details:	
Training set size	800,000
Validation set size	100,000
Epochs	5
Batch Size	128
L2 weight decay	alpha: 0.4
Optimizer	Adam
Learning rate	10^{-3}
Bi-LSTM Hidden state size	200
Bi-LSTM Inter-layer dropout probability	0.5
Bi-LSTM Number of layers	2
Input dropout probability	0.5
Output dropout probability	0.5

Character representation convolution Number of filter sizes	1
Character representation convolution Filter sizes	{1}
Character representation convolution number of filters	128
ELMO pre-trained embedding dimension	1024
GloVe pre-trained embedding dimension	50

Table 3: Details for Final Trained Model

A patience level is set to 1 so that the model will stop training at the second epoch to prevent overfitting since in the configuration file the first epoch starts from 0 and the second epoch is 1. The table below will show the training and validation metrics of the trained model.

	Precision	Recall	F1 score	Loss
Training	0.99677	0.99625	0.99651	14.427
Validation	1.0	1.0	0.99999	-0.00011

Table 4: Training and validation metrics

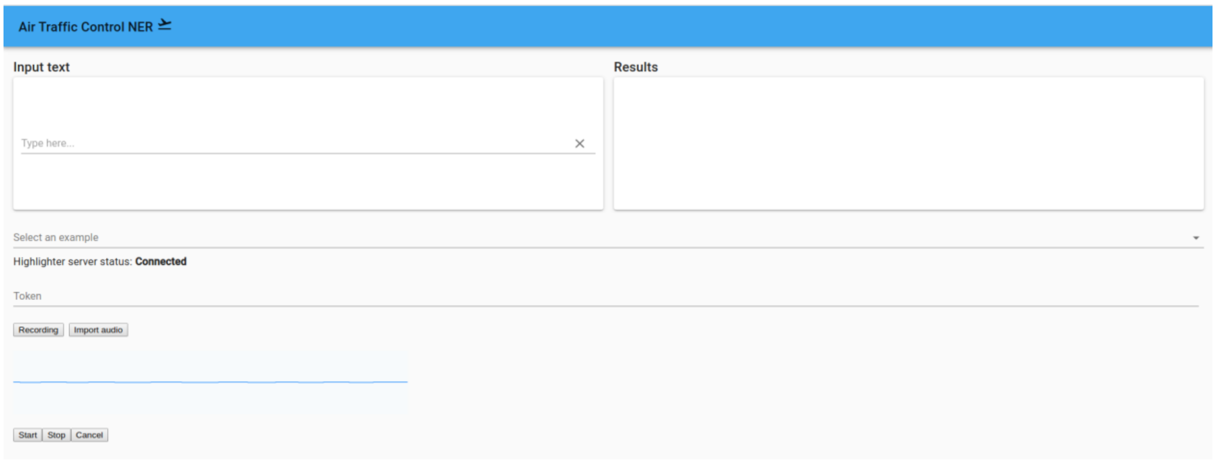
3.5 Speech to text recognizer

A speech to text module was integrated into the system so that it can simulate speeches made by the ATC. It is an automatic speech recognition (ASR) system that was implemented by one of the researches in AI speech lab. AI speech lab is a system that performs speech recognition for speech-to-text. The online system from AI speech lab is able to perform speech recognition in real-time. Data can be streamed to the system by using a microphone input to the website. After that, the audio data will be decoded, and the transcription will be done in real-time. To connect to the online ASR system, a specific web socket address must be used. A token is also required to connect to the online ASR system. An example of a web socket address is shown below.

```
ws://HOST:PORT/client/ws/speech?content-type=audio/x-
raw,+layout=(string)interleaved,+rate=(int)44100,+format=(string)S16LE,
+channels=(int)1?token=<YOUR_ACCESS_TOKEN>
```

3.6 Demo

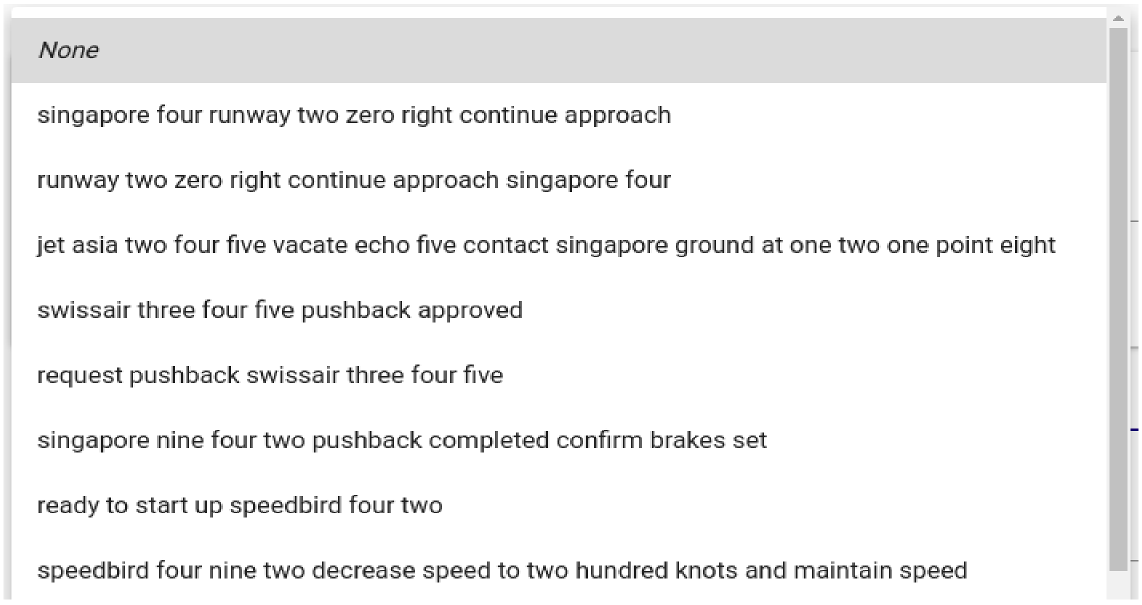
The demo of the keyword and name entity recognition of ATC text was implemented in a web application. It was developed in React JS [34], a popular framework of Javascript. It is hosted locally. The website can allow the users to input their ATC text either by typing the ATC conversations on the website or through the use of the speech to text button. The speech to text will work only when a valid token is place inside the token input box. The token was given by one of the researches in AI Speech lab.



Highlighter server status: Connected

Figure 23: Screenshot of web application

Users are able to select some examples if they do not know what to type on the website. The below figure shows some of the examples that an air traffic controller will speak to the pilots or vice versa.


None

singapore four runway two zero right continue approach

runway two zero right continue approach singapore four

jet asia two four five vacate echo five contact singapore ground at one two one point eight

swissair three four five pushback approved

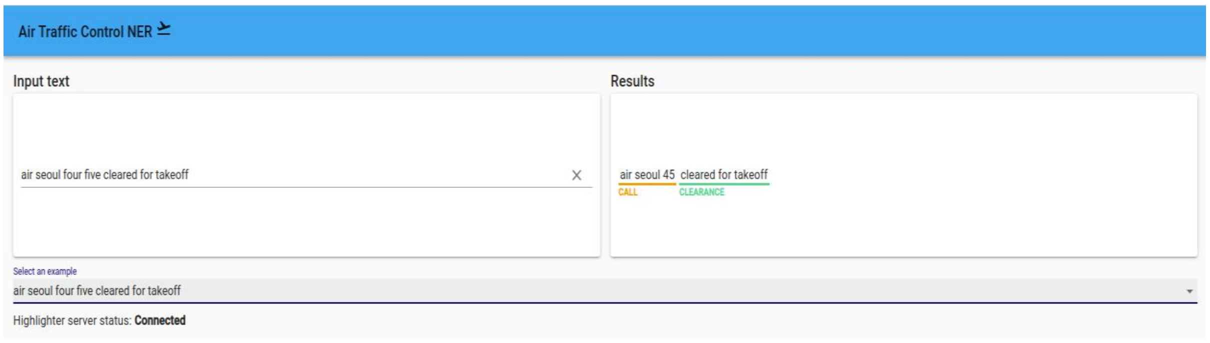
request pushback swissair three four five

singapore nine four two pushback completed confirm brakes set

ready to start up speedbird four two

speedbird four nine two decrease speed to two hundred knots and maintain speed

Figure 24: Examples of ATC text in website


Air Traffic Control NER

Input text

air seoul four five cleared for takeoff

Results

air seoul 45 cleared for takeoff

CALL CLEARANCE

Select an example

air seoul four five cleared for takeoff

Highlighter server status: **Connected**

Figure 25: Input ATC text with result

Based on Figure 25, the user can input ATC text on the left and the results on the right will show the highlighted name entities with different colours. In addition, when words of numbers are being typed into the website, they will be converted to numbers. For instance, when air seoul four five is typed into the input text box, it will be converted to to air seoul 45 in the results box. This will be easier for the users to read.

Users are able to know whether the website detects their voice by observing the waveforms of the visualizer. The figure below will show a random waveform on the visualizer.

Figure 26: Speech visualizer

Chapter 4

Conclusion and future work

4.1 Conclusion

This project was done to improve the number of name entities and keywords of ATC conversations that will be detected in the demo website. A grammar file which uses context free grammar contains grammar rules of ATC conversations. Through the use of this grammar file, sentences of ATC conversations will be generated. These sentences will be annotated in a BILOU format. After that, a model will be trained using the configuration file provided by Allennlp. Google Colab will be used to train the model and lastly, the model will be deployed in the demo website so that users can see the highlighted name entities and keywords in real time. In addition, a speech to text function will allow users to speak through a microphone and they can observe their speech waveforms and the annotated name entities and keywords on the website.

4.2 Future Work

There are some improvements that could be made in this project. Here are some of the suggestions:

- Currently, the dataset contains conversations between ATC and pilots. Planes are not only the ones that are on the airports. There are other ground vehicles that could be present in the airport runways. For instance, baggage collectors could be using one of the airport runways to transport baggage from one area to another. Hence, ATC text should include conversations between ground vehicle operators and ATC.
- Improve the annotation time for ATC text. Currently, the annotation time for annotating the name entities in BILOU format takes around two days for the annotation of 1 million sentences. This is likely to increase as there will be more

name entities and grammar rules that will be added into the grammar file. To solve this problem, the grammar file can be used to generate name entity tags using grammar rules so that time will be not spent on annotating the generated sentences.

- There are many ways an air traffic controller can give instructions to the pilots. To fully understand ATC speeches, it will be good to listen to live ATC speeches spoken by the air traffic controllers themselves. A more accurate model can be trained if real world datasets are used.

References

[1] Patty, A., Fatal consequences of miscommunication between pilots and air traffic controllers. Retrieved from https://www.smh.com.au/business/workplace/the-fatal-consequences-of-miscommunication-between-pilots-and-air-traffic-controllers-20160928-grq1d9.html, 2016

[2] Tyagi S., Mishra G. S., Statistical Analysis of Part of Speech (POS) Tagging Algorithms for English Corpus, 2015

[3] Li J., Sun A., Han J., Li C., A Survey on Deep Learning for Named Entity Recognition, 22 December 2018

[4] Naskar, A., “Extract Custom Keywords Using NLTK POS Tagger in Python.” Extract Custom Keywords Using NLTK POS Tagger in Python, 1 Oct. 2018, https://www.thinkinfi.com/2018/10/extract-custom-entity-using-nltk-pos.html.

[5] Ravi, B., “Named Entity Recognition Using Spacy.” Entity Extraction-Demistifying RasaNLU-Part 3, 14 July 2018, https://hackernoon.com/entity-extraction-demistifying-rasanlu-part-3-13a460451573.

[6] Gupta, M., A Review of Named Entity Recognition (NER) Using Automatic Summarization of Resumes. Retrieved from https://towardsdatascience.com/a-review-of-named-entity-recognition-ner-using-automatic-summarization-of-resumes-5248a75de175, 10 July 2018

[7] Gupta, S., Named Entity Recognition: Applications and Use Cases. Retrieved from https://towardsdatascience.com/named-entity-recognition-applications-and-use-cases-acdbf57d595e, 10 February 2018

[8] Arora S. J., Singh R. P., International Journal of Computer Applications (0975 - 8887) Volume 60 – No.9, Automatic Speech Recognition: A Review, December 2012

[9] Brownlee, J., Supervised and Unsupervised Machine Learning Algorithms. Retrieved from https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/, 12 August 2019

[10] Mishra, S., Unsupervised Learning and Data Clustering. Retrieved from https://towardsdatascience.com/unsupervised-learning-and-data-clustering-eeecb78b422a, 21 May 2017

[11] Lipton, Z. C., Berkowitz, J., & Elkan, C., A Critical Review of Recurrent Neural Networks for Sequence Learning, 5 June 2015

[12] Yao M., Chen Y., Understanding Hidden Memories of Recurrent Neural Networks, October 2017

[13] Hochreiter, S., Schmidhuber, J., Long Short-Term Memory. Neural Computation 9(8): 1735-1780, November 1997

[14] Mandelbaum, A., Shalev A., Word Embeddings and Their Use in Sentence Classification Tasks, 27 October 2016

[15] Peters, M.E., Neumann M., Iyyer M., Gardner M., Clark C., Lee K., Zettlemoyer L., Deep contextualized word representations, March 2018

[16] Purohit, Karan. “Learn How to Build Powerful Contextual Word Embeddings with ELMo.” Medium, Saarthi.ai, 4 June 2019, medium.com/saarthi-ai/elmo-for-contextual-word-embedding-for-text-classification-24c9693b0045.

[17] Batista D. S., Named-Entity Recognition based on Neural Networks, 22 October 2018

[18] Putera F., Keyword and Named Entity Recognition on Online Text Data, 2019

[19] Jason P.C. Chiu and Eric Nichols. Named Entity Recognition with Bidirectional LSTM-CNNs. Transactions of the Association for Computational Linguistics, 4:357–370, December 2016.

[20] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL2003 Shared Task: Language-Independent Named Entity Recognition. In Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003, pages 142–147, 2003.

[21] Wan YH., A Novel Approach for English Phonetic Alphabet in Wireless Communication, 24 July 2013

[22] Pliso A., Non-Standard Phraseology in Aviation English, August 2014

[23] Francetić I., Radiotelephony communications 1 Handbook, University of Zagreb Faculty of transport and traffic sciences, 6 May 2013

[24] Arnot, M., What Do Runway Numbers Mean? Retrieved from https://thepointsguy.com/guide/what-do-runway-numbers-mean/, 10 June 2018

[25] Mjåtveit, Stein. “Learn to Talk like a Pilot: Clearances.” OSM Aviation Academy, 4 June 2018, www.osmaviationacademy.com/blog/learn-to-talk-like-a-pilot-part-3-clearances, 4 June 2018

[26] Tu, Ho Dac, et al. “The Next Generation Air to Ground Communication System for Air Traffic Control.” IEEE/ACES International Conference on Wireless Communications and Applied Computational Electromagnetics, 2005., May 2005, doi:10.1109/wcacem.2005.1469754.

[27] Malik M. K., Sarwar S. M., (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 7, Named Entity Recognition System for Postpositional Languages: Urdu as a Case Study, 2016

[28] Thia, J. M. X., Keyword and Named Entity Recognition on Air Traffic Control (ATC) data, 2019

[29] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global Vectors for Word Representation. In Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, 2014.

[30] Allauzen C., Riley M., Schalkwyk J., Skut W., Mohri M., OpenFst: A General and Efficient Weighted Finite-State Transducer Library, July 2007

[31] Roark B., Sproat R., Allauzen C., Riley M., Sorenson J., Tai T., The OpenGrm open-source finite-state grammar software libraries, 8-14 July 2012

[32] Paszke A., Gross S., Massa F., Lerer A., Bradbury., Chanan G., Killeen., Lin Z., Gimerlshein N., Antiga L., Desmanison A., Köpf A., Yang E., DeVito Z., Raison M., Tejani A., Chilamkurthy S., Steiner B., Lu F., Bai J., Chintala S., PyTorch: An Imperative Style, High-Performance Deep Learning Library, 2019

[33] Gardner M., Grus J., Neumann M., Tafjord O., Dasigi P., Liu N., Peters M., Schmitz M., Zettlemoyer L.. AllenNLP: A Deep Semantic Natural Language Processing Platform. 2017.

[34] Xing Y., Huang JP., Lai YY., Research and Analysis of the Front-end Frameworks and Libraries in E-Business Development, 23 February 2019