Radboud University Nijmegen

Faculty of Social Science

Classification of Readback Errors

An Implementation For Aviation English

Thesis BSc Artificial Intelligence

Author:
Nur Atiqah Zakiah
Abdul Khaliq
s1004653

Supervisor:
Dr. Franc Grootjen

Second Reader:
Dr. Umut Güçlü

22 July 2020

Contents

Abstract

Readback errors in English Air Traffic Control speech pose a significant safety risk. The National Aeronautics and Space Administration (NASA) and Federal Aviation Administration (FAA) have published various reports illustrating communication errors as prominent factors resulting in aviation incidents. While communication errors exist in a broad range, readback and hearback errors will be the main focus in this thesis. In an attempt to reduce readback or hearback errors, this research looked at the possibility of automating the hearback process where controllers are required to verify readbacks made by pilots. The task of classifying readbacks as 'Correct', 'Incomplete' or 'Wrong' was seen as analogous to a sentence matching task, where the relationship between instruction from a controller and readback from a pilot was determined. A processing pipeline was introduced as 'The Proposed Scheme' and Natural Language Processing techniques and Convolutional Neural Network architectures were explored. The best implementation of the proposed scheme had a 78.2% recall rate, higher than the most recent reported recall rate of human controllers which stood at 77.8%.

1 Introduction

Controllers in the domain of Air Traffic Control (ATC) manage aircrafts on the ground and through controlled airspaces. A big challenge in introducing automation to the ATC domain is the intensive use of voice radio communication. Pilots and controllers rely primarily on radio transmissions to communicate and it is critical that both parties understand each other accurately. Controllers ensure the safety of aircraft within their jurisdiction by issuing clearances and advisories to pilots. Controllers also verify readbacks from pilots, ensuring that pilots repeat clearances or advisories fully and correctly and then monitor aircraft movements to confirm pilots' compliance to issued commands. However, errors do occasionally occur in these processes resulting in aviation incidents. Aviation incidents are defined by the Convention on International Civil Aviation in Annex 13 (International Civil Aviation Organization, 1994), as events associated with the operation of an aircraft that affect or could potentially affect its safety.

Various researchers have warned about the dangers of miscommunication in ATC, it was found that at least 2,000 people have died in aeroplane crashes in the last 45 years, in which communication errors were a strong factor (Patty, 2016) and language-related communication problems tend to arise during nonstandard and emergency situations (Gontar et al., 2017). The latest update to the International Air Transport Association's (IATA) 20-Year Air Passenger Forecast projected that passenger numbers could double to 8.2 billion in 2037 (International Air Transport Association, 2018). The increase in flight volume contributes to added pressure and stress experienced by controllers and pilots, as well as frequency congestion, these could lead to human errors which are already a significant factor in miscommunication. Technical reports from Federal Aviation Administration (FAA) and National Aeronautics and Space Administration (NASA), including the Aviation Safety Reporting System (ASRS) report, have stated that communication errors such as misunderstandings are a prominent factor resulting in aviation incidents (Billings and Cheaney, 1981; Cardosi et al., 1998; Schroeder et al., 2007). Moreover, Eurocontrol has identified readback or hearback errors as the most common error in ATC speech (Van Es, 2004). While communication errors exist in a broad range, this research will therefore focus on the classification of readback errors.

Verifying readbacks constitutes a significant part of an ATC controller's responsibilities. This process may also be referred to as making a hearback, and here controllers ensure pilots repeat instructions properly and to correct any mistakes a pilot may have made in a readback. A readback error occurs when a pilot fails to repeat the instruction from a controller fully and accurately. Errors in readback or hearback could lead to situations involving altitude deviation, runway incursion, airborne conflict or operational deviation among many others. The most frequent consequence of hearback errors was found to be prolonged loss of communication. Additionally, it should be highlighted that a single error could lead to multiple consequences. Often, it occurred that when an error leads to a wrong aircraft accepting an instruction from the controller, this was followed by the wrong aircraft deviating from its appropriate altitude (Es, 2004). When making a hearback, in the best-case scenario: the instruction and readback are exactly the same sentences. In the worst-case scenario: there is no readback. However in reality, most readbacks fall between these two extremes, making it difficult to verify if readbacks are correct. In a series of studies in the 1990s, researchers found that ≤ 1 % of readbacks were incorrect and of those approximately 40-45% were not corrected by controllers (Burki-Cohen, 1995; Cardosi and Han, 1997). In the 2000s, researchers reported readback errors rates were at 6% and out of 723 readback errors 92% were not corrected (Prinzo et al., 2006). More recently, the readback error rate was again found to be $\leq$

1% and the hearback error rate was reported to be 22% (Lennertz, 2017). Automating the process of hearback i.e. readback error classification would involve several areas of natural language processing (NLP) such as speech recognition and semantic processing. Additionally, the situational awareness and readiness of both controllers and pilots are imperative.

1.1 Research Question

Classifying readback is analogous to sentence matching or paraphrase identification that is applied to ATC speech. In sentence matching, it is critical to identify the relationship between two sentences. It is important to be able to appropriately model the relationship and compare the textual similarity. Similarly in classifying readback, it is necessary to identify the relationship between the instruction given by controllers and the readback provided by pilots. Here it is also relevant to look at the textual similarities especially in semantics. Many algorithms have been proposed for sentence matching, some take advantage of word embeddings to express words as vectors in semantic space (Kenter and de Rijke, 2015) while others have utilized linguistic analysis (Mihalcea et al., 2006), lexical matching (Islam and Inkpen, 2008), and artificial neural networks such as deep neural networks (Agarwal et al., 2018) and convolutional neural networks (CNNs) (Hu et al., 2014)). In spite of the double-check that readbacks afford, human errors tend to be difficult to notice because controllers are susceptible to inattentional blindness, attentional blink, working memory overload and disruption of memory consolidation (Xing and Bailey, 2005). In hopes to reduce the mental load controllers endure, by automating the hearback process, the research question is as follows:

Is it possible to build an effective classifier for readbacks in English ATC speech through the use of Natural Language Processing (NLP) techniques and Convolutional Neural Networks (CNNs)?

It should be noted that the outcomes of this research is dependent on speech recognition technologies which are robust and capable of transcribing conversations between pilots and controllers accurately.

1.2 Structure

This bachelor thesis can be broken down into several components. The first stage looked into computationally modelling the classification of readbacks in English ATC speech. A suitable scheme for readback classification was then developed, based on methods used in previous research, and proposed. The proposed scheme was intentionally developed to be general enough to allow variations in implementation so that various configurations can be evaluated to identify the most efficient classifier.

The second stage entails the collection of data. The aim of the data collection stage was to create a dataset of textual Instruction-Readback (I-R) pairs with the corresponding class label for the relationship within the pair of texts - Correct, Wrong or Incomplete.

The third stage was to implement relevant models that were identified in the literature survey. In this study, two relevant models were identified. The corresponding research were studied and the models proposed in those were replicated and then adjusted to fit the proposed scheme. Additionally, an original novel model was developed by incorporating features from existing research in a new arrangement. With that, the dataset was then used in the this stage to train the models implemented.

The fourth stage of this bachelor thesis addressed the evaluation of the various models that were implemented. In this section the research question *'Is it possible to build an effective classifier for readbacks in English ATC speech through the use of NLP techniques and CNNs? '* was addressed through the efficacy of the proposed scheme. As such it was necessary to decompose the research question into two parts - 1) which implementation of the proposed scheme should be adopted and 2) with this implementation, is it possible to automate the readback classification? These questions were addressed in the Experiments and Results section.

2 Background

This chapter will give some insights into phraseology in Aviation English and guidelines with regards to readback and hearback, as well as explore existing literature on the problem of readback classification.

2.1 Aviation English Phraseology

Aviation English, a semi-artificial sublanguage based on English, serves as the de facto international language of civil aviation and plays a dominant role in communication between pilots and air traffic controllers especially in international contexts. English and Aviation English differ in terms of phonology, lexis and syntax (Breul, 2013). International Civil Aviation Organization (ICAO), which was founded by the United Nations to set up international standards in aviation, developed a series of diction and pronunciation standards that have been formulated in a bid to improve language norms and terminology standards in Aviation English.

> ***'Swedestar 05Z line up RWY 31' 'Lining up RWY 31 Swedestar 05Z'***
> Here, the callsign would be pronounced *'Swedestar zeero-fife-zulu'* and the runway id would be pronounced *'runway tree-wun'.*

The diction and pronunciation standards are specified in ICAO Annex 10 (International Civil Aviation Organization, 2001). It includes the pronunciation of alphabets, numbers, times as well as call signs and is being improved continuously, this facilitates pronunciation for non-native English speakers. Additionally, Aviation English features various standard words and phrases to convey meanings often expressed differently in natural conversation. For example, *"affirm"* is used in place of *"yes"*, and *"over"* is used to express the end of a message and expectation of a response. Syntax also differs between English and Aviation English, the most significant difference lies in the deletion of parts of speech in order to maximise brevity - determiners, prepositions and pronouns are often omitted. For example *"resume own navigation"* is said in place of *"resume your own navigation"* or *"climb 150"* usually means *"climb to flight level 150"*. Furthermore, controllers and pilots are trained to strictly abide by the ICAO standards to reduce the likelihood of aviation incidents. Despite these measures, communication errors persist and readback/hearback problems still exist. These issues are also a significant worry as the aviation industry continues to grow, this added pressure on controllers and pilots increases the likelihood of mistakes being made.

2.1.1 Defining Readback Requirements

Readbacks provide controllers with the opportunity to verify that pilots have understood instructions and clearances correctly. Pilots should provide a readback that is clear and

complete so that it can be understood by the controllers and allow controllers to rectify any misunderstandings. Requirements for readbacks are stringent due to its direct relation to the serious implications that a possible misunderstanding in the transmission and receipt of ATC clearance and instruction can cause (International Civil Aviation Organization, 2001). It was identified that readback and hearback errors could result in one or more of the following types of events: altitude deviation, less-than-standard separation, wrong aircraft accepting clearance, operational error or heading/track deviation (Cardosi et al., 1998).

ICAO (Civil Aviation Authority, 2016) and CAA guidelines (Civil Aviation Authority, 2015) established that instructions with information specifying any of the items below should be included in the readback and should always include the aircraft callsign: (1) Taxi/Towing Instructions, (2) Level Instructions, (3) Heading Instructions, (4) Speed Instructions, (5) Airways or Route Clearances, (6) Approach Clearances, (7) Runway-in-Use, (8) Clearance to Enter, Land On, Take-Off On, Backtrack, Cross, or Hold Short of any Active Runway, (9) SSR Operating Instructions, (10) Altimeter Settings, including units when a value is below 1000 hectopascals, (11) VDF Information, (12) Frequency Changes, (13) Type of ATS Service and (14) Transition Levels.

Controllers are obliged to ask for a readback if it was not received and pilots should not use the terms "Roger", "Wilco", or "Copies" in place of a complete and correct readback.

2.2 Related Work

Readback verification has the potential to significantly reduce the frequency of communication errors. While, English serves as the de facto international language for civil aviation, more research about readback verification in Chinese ATC was found to be publicly available. With that, this thesis aims to fill the gaps in existing research with regards to readback verification in Aviation English. In this section, existing research relevant to the task of readback classification will be reviewed. Additionally, given the recent trend of using CNNs which were originally designed for computer vision on NLP tasks, existing works in relevant NLP domains like Sentence Matching, and Sentence Classification will also be reviewed.

2.2.1 Readback Classification in Existing Literature

A literature survey gave insight into methods that have already been utilized to classify readback errors. Although the main goal of Chen et al. (2017) was to study the feasibility of using automatic speech recognition technologies in ATC, readback detection tasks were used as a validation measure. They developed a Semantic Meaning Extraction Algorithm to identify instruction phrases and its parameters, and a Pattern Matching Algorithm to identify aircraft call signs. The techniques used to build these algorithms were not divulged in their research article however in a small case study consisting of only 199 hold-short instructions, they reported 86% accuracy in detecting readback errors.

Jia and Cheng (2018) sought to solve the task of readback classification through deep learning, they developed a semantic checking model using Long-Short-Time Memory networks (LSTM). Their proposed architecture boosts robustness by introducing a mean-pooling layer to exploit all the information in the hidden layers, a multilayer perceptron (MLP) in place of a similarity function. The output of the MLP layer was then passed to a K-Nearest Neighbour classifier. In order to validate this model, the researchers

created a corpus of readbacks in Chinese ATC which consisted of 2442 Instruction and Readback (I-R) pairs with 1,326 positive samples and 1116 negative ones. The negative samples were deliberately designed according to an investigation of communication problems in ATC speech in order to overcome the class imbalance as readback errors do not regularly occur. This architecture was able to produce an impressive accuracy rate of above 92%.

Subsequently, Cheng teamed up with Jia again in another bid to solve the readback classification task. In this research (Cheng et al., 2018), another architecture was proposed - it had two channels of a one-layer CNN to process the instruction and the readback sentences. It was claimed that with one-layer, a CNN was able to learn the semantics of the I-R pairs. The pairs were then classified according to a matching vector which was a concatenation of the semantic vectors of the instruction and the readback and the cosine similarity. The Chinese Civil Radiotelephony Communication (CCRC) dataset was also developed in this research to assess the performance of the proposed method. The dataset was based on the ATC recordings between controllers and pilots, and radiotelephony training books. This dataset included 3800 pairs and distinguished between readbacks labelled into seven classes: correct, heading information error, runway information error, call sign information error, altitude information error, and partial information loss with 1300 pairs in the correct category and 500 in each error category. It was concluded that this method was able to achieve accuracy rates above 95% on the CCRC dataset when readbacks were doubled i.e. represented twice, as this strategy strengthened the semantics and allowed CNNs to extract better representations. This is an improvement from their previous research (Jia et al., 2018), as this method yielded better results with a simpler architecture.

2.2.2 Sentence Matching and Sentence Classification

CNNs have become an increasingly popular method for various NLP tasks and have recently demonstrated impressive performance on text classification (Hu et al., 2014; Johnson and Zhang, 2015; Severyn and Moschitti, 2015). Collobert et al (2011)were one of the earliest to model sentences using CNNs, forming sentence representations by applying convolutions on windows that slide over a sentence and using max-pooling. Later, Kalchbrenner (2014) stacked convolutional layers which used sliding windows to split sentences into n-grams, with higher layers extracting more abstract features by combining information from lower layers thus achieving multi granularity, and introduced k-max-pooling which extracted the k features with the highest values. These research showed that CNNs are able to learn the structures and meanings of long sentences regardless of the positions of its elements.

The task of readback classification can be seen as analogous to sentence matching and classification in NLP as sentence pairs that semantically match are classified as correct and sentence pairs that do not as incorrect.

Existing research has demonstrated CNNs performance on such tasks as well as illustrated the various architectures employed. Researchers (Hu et al., 2014) came up with two convolutional architectures ARC-I and ARC-II for sentence matching, both architectures combined learning sentence representations and training the classifier. ARC-I adopted a Siamese network whereby two CNNs, worked in parallel to learn the representation for each sentence, were connected to an MLP for classification. ARC-II, on the other hand, allowed the two sentences to interact during the process sentence representation formation by applying sliding windows over both sentences and modelling

all possible combinations through 1D convolutions. However, there was no significant difference in performance for a paraphrase identification task, ARC-I had an accuracy of 69.6% and ARC-II had an accuracy of 69.9%.

Kim (2014) used word vectors obtained from an unsupervised neural language model, word2vec (Mikolov et al., 2013), to train a one-layer CNN for sentence classification. It was concluded that this simple model achieved impressive results in a variety of tasks despite minimal tuning.

Additionally, given the successes of CNNs on sentence classification tasks, Zhang & Wallace (2017) looked into sensitivity of CNNs performance with regards to its configurations. Their research specifically analysed one-layer CNNs to identify important and comparatively inconsequential design decisions, and discovered that sentence representation, filter region size and number of feature maps are influential hyperparameters requiring tuning.

3 Method

In this section, a scheme is proposed to solve the task of readback classification in English ATC speech. In order to evaluate the effectiveness of that scheme, a dataset - the Readback Dataset, was built and the scheme was implemented in three different models with varying architectures. The models implementing the three different architectures serve as independent variables for the experiments conducted to answer the research question. While the implementation stage in this section provides a glimpse of how well each model performs, the experiment and results explores each model's efficacy in detail.

3.1 Proposed Scheme For Readback Classification

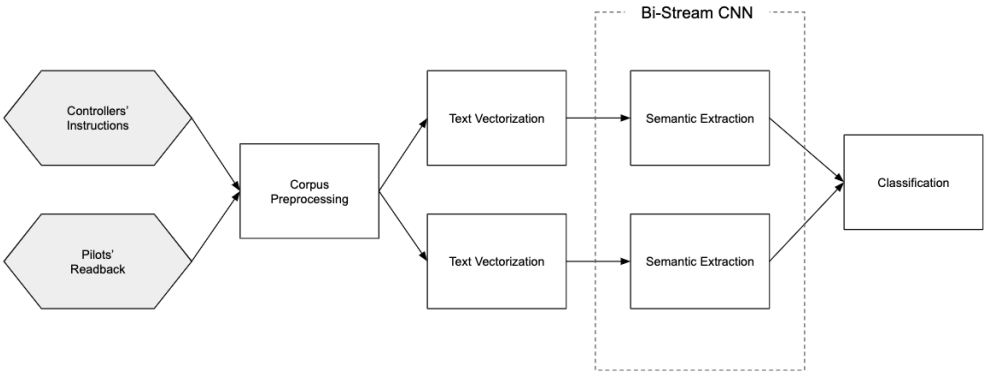


Figure 1: The Proposed Scheme for Readback Classification

Taking inspiration from previous research tackling readback classification (Cheng et al., 2018; Jia et al., 2018, 2017), the scheme handles I-R pairs in textual format and features a bi-stream CNN network as well. The scheme involves processing the controller's instructions and the pilot's readbacks which is fed into a bi-stream CNN to extract the sentences' semantic features and then given as input into a classifier to map the relationship between the two sentences. The proposed scheme is visualized above.

Corpus Preprocessing. In Corpus Preprocessing, the transcribed instructions and

the readbacks will be cleaned and segmented. For efficiency, since the same transformations will be applied on the instructions and readbacks, both are concatenated to form a corpus and passed along a preprocessing pipeline.

Text Vectorization. This step uses embedding techniques trained on the entire corpus to capture all unique words present in the instructions and readbacks. The segmented text is transformed to a corresponding vector so that it can be used as input into the CNN layer.

Semantic Extraction. As seen in previous research, CNNs have demonstrated impressive performance on NLP tasks (Johnson and Zhang, 2015; Zhang and Wallace, 2017). Beyond being popular for sentence classification, there has been strong empirical evidence suggesting shallow CNN architectures in particular (Cheng et al., 2018; Kim, 2014; Zhang and Wallace, 2017) to be useful in extracting semantic meaning due to their feature detection capabilities. CNNs are especially relevant for this task as the length of the instructions and readbacks are relatively short and the word presence is more important than word order. In this proposed scheme, a bi-stream CNN architecture would be used to extract semantic features of each sentence. The vectorized instruction and readback sentences would be fed into a bi-stream CNN. The output of the Semantic Extraction step would be a concise semantic vector of each sentence.

Classification. The proposed scheme incorporates training the bi-stream CNN in Semantic Extraction and an MLP classifier together. Research showed that combining the steps of sentence modelling and classification into one network resulted in better performance than training them separately (Yin and Schütze, 2015). The semantic vectors generated by Semantic Extraction will be given as input into an MLP which consist of a hidden layer and a final layer with a softmax to predict the class label representing the relationship between the I-R pair.

3.2 Building Readback Dataset

Given, there was no dataset built for the purpose of classifying I-R pairs in English ATC speech publicly available, much less one in textual format. A corpus was created to train and test models built based on the proposed scheme in order to evaluate its efficacy. The corpus is based on transcriptions of ATC speech recordings between pilots and air traffic controllers obtained from the Air Traffic Control Communication (ATCC) corpus.

3.2.1 Data Collection

The ATCC corpus was published by Šmídl et al (2019) from University of West Bohemia, Department of Cybernetics for the research project "Intelligent technologies for improving air traffic security (IT-BLP)". It contained communication from the Air Navigation Services of the Czech Republic in Jenec and some from the Lithuanian and Philippines airspaces.

TRS Transcriber files containing the transcribed ATC speech from the ATCC corpus were obtained from the LINDAT-Clarin repository published online by Smidl and Lubos. The ATCC corpus contained communications from three different domains - tower, approach and area control. The tower domain is responsible for takeoff, landing and landing standby correspondences, the approach domain is responsible for communicating with aircrafts approaching the airspace to land and the area control domain is responsible for communication during overflights and cruises. Instructions from controllers

and the corresponding readback from pilots were extracted from the TRS files to form I-R pairs. Extraction of complete instructions and readbacks from the transcribed TRS files proved to be a time-consuming task, as many sentences were not fully transcribed. Not all utterances could be understood due to radio noise, unclear pronunciation from the speakers, and fast speech. Additionally, utterances were often cut off before the sentence was completed. Since the ATCC corpus contained more data from the area control domain, this translates to more I-R pairs about level, speed and contact instructions as aircrafts move from one airspace to another. To ensure equal representation, 250 random I-R pairs were extracted for each domain summing up to 750 I-R pairs in total. In order to ensure random pairs, the TRS files were first divided into the respective domains, then a short AppleScript program was written to assign random 3-digit numbers at the start of the filename. From there, the files in each domain were sorted in ascending order and combed to extract the instructions and corresponding readbacks until 250 I-R pairs were extracted.

3.2.2 Data Labelling

I-R pairs are labelled into "Correct", "Wrong" and "Incomplete" classes according to the degree of understanding. Correct readbacks demonstrate full understanding, a wrong readback shows misunderstanding and an incomplete readback falls somewhere in between. An incomplete readback does not allow the controller to distinguish if the pilot understood the instruction fully or not because pieces of information are missing. To label the extracted I-R pairs, the guidelines specified by ICAO in Annex 10 and CAA in CAP 493 were referred to. ICAO specified that pilots must always include the call sign of the aircraft when making readbacks in order to avoid any possible confusion. The figure below obtained from Chapter 5 of Annex 10 (International Civil Aviation Organization, 2001) illustrates how call signs may be abbreviated.

	Type (a)		**Type (b)**	**Type (c)**
Full Call Sign	ABCDE	Airbus ABCDE	Rushair BCDE	Rushair 1234
Abbreviated Call Sign	ADE or ACDE	Airbus DE or Airbus ABDE	Rushair DE or Rushair BDE	No Abbreviated Form

Table 1: Examples of Full Call Signs and Abbreviated Call Signs.

Most airlines call signs come under type (c) and have no abbreviated form and specifying only the carrier or the flight number is not acceptable. Abbreviations are only tolerated if the controller has used the abbreviation or satisfactory communication has been established and the abbreviation is not likely to cause confusion. In all other cases, readbacks made with abbreviated call signs will be categorized as wrong. Also, readbacks made without the aircraft's call sign are classified as incomplete. In addition to ICAO and CAA guidelines, several behaviour patterns in ATC speech identified in previous literature (Chen et al., 2017) were also taken into consideration as similar patterns were observed in the ATCC corpus. Different order of phrases between instruction and readback, the use of words semantically equivalent to "follow" such as "after" or "behind", omission of "left turn" or "right turn" as long as the heading vector was correctly specified were all tolerated and did not contribute to an error in readback making. A decision-making chart was developed to visualize how the I-R pairs should be categorized into each class.

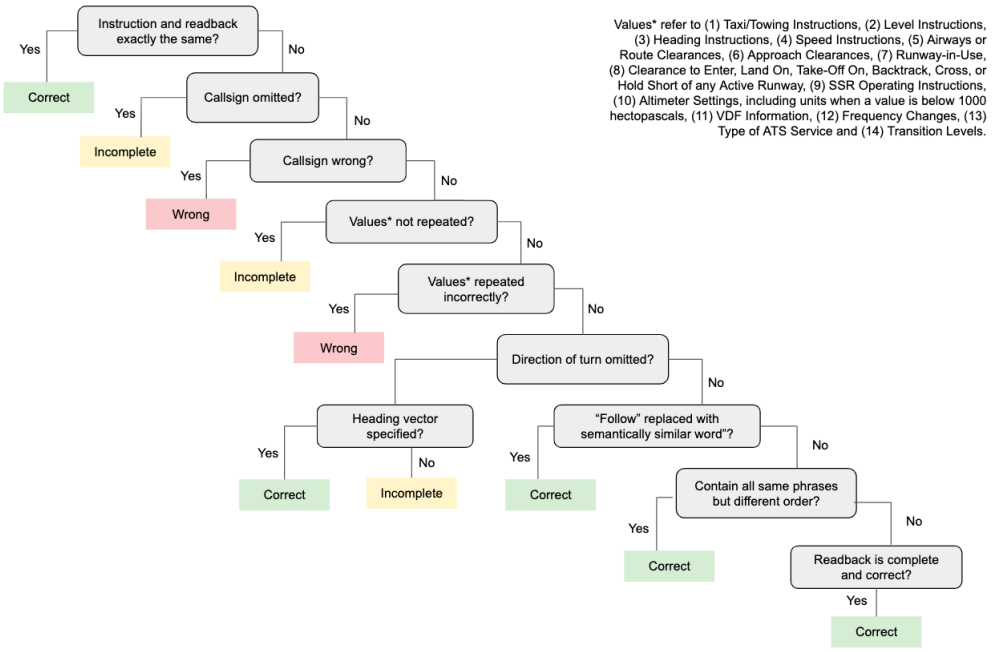


Figure 2: Decision Tree to Classify Readbacks

Out of 250 I-R pairs in the tower domain, 181 pairs were correct, 26 wrong and 43 incomplete. For the approach domain, 187 pairs were correct, 21 wrong and 42 incomplete. Similarly for the area control domain, 209 pairs were correct, 22 were wrong and 19 incomplete.

Class/Domain	**Tower**	**Approach**	**Area Control**
Correct	181	187	209
Wrong	181	21	22
Incomplete	43	42	19

Table 2: Counts of samples in each domain

3.2.3 Data Augmentation

Data augmentation offers a useful way to significantly increase the number of samples for training the readback classification models without extracting more I-R pairs from the ATCC corpus. Beyond expanding the dataset, data augmentation also changes the makeup of the data and is often used to overcome class imbalances. Increasing training set size and representation of minority classes improves model performance and increases generalizability. Additionally, augmentation was recognised as one of the best practices with regards to convolutional neural networks applied to document analysis (Simard et al., 2003). Since the Readback Dataset built so far contains 750 samples with 77% of those classified as correct, data augmentation was used to avoid overfitting through the increase in training set size and reduction of class imbalance. Moreover, the dataset should be augmented to reflect the flexible nature of Aviation English given that there are arrays of acceptable and unacceptable readbacks for a single instruction.

Various techniques to augment text data exist but replacing words or paraphrasing were not appropriate for this dataset as swapping keywords with synonyms may violate the phraseology of Aviation English. Thus, word shuffling was adopted as a different order of phrases between instruction and readback is tolerated. However, sentences in the Readback Dataset were not annotated, this made automatic shuffling problematic and the resulting sentences implausible i.e. sentence does not make sense or is not likely to occur. As such, the sentences in the dataset were shuffled manually.

The sentences were augmented in such a way that there will be 3 correct, 1 incomplete and 1 wrong readbacks for each unique instruction. Besides shuffling, some relevant words were added or removed. For example "flight level" or "climb level" was sometimes shortened to "level". To come up with manually augmented readbacks, inspiration was drawn from CAP 49 and phrases often used in extracted readbacks. For augmentation of incomplete and wrong readbacks in the minority classes, common mistakes found in extracted readbacks were referred to. With that, 750 samples were expanded to 3,750 samples and the percentage of incomplete and wrong readbacks almost doubled from 22% to 40%. The figures below illustrate how a single original sample in the first row was expanded into five samples.

Instruction	**Readback**	**Class Label**	**Domain**
CIG 1642 climb flight level 340	flight level 340 CIG 1642	Correct	ACCU
CIG 1642 climb flight level 340	climbing flight level 340 CIG 1642	Correct	ACCU
CIG 1642 climb flight level 340	CIG 1642 flight level 340	Correct	ACCU
CIG 1642 climb flight level 340	flight level 340 CIG	Incomplete	ACCU
CIG 1642 climb flight level 340	flight level 340 CIG 1632	Wrong	ACCU

Table 3: Data Augmentation of an originally correct sample (first row)

Instruction	**Readback**	**Class Label**	**Domain**
Lufthansa 6TU climb to FL 300	6TU climbing FL 300	Incomplete	ACCU
Lufthansa 6TU climb to FL 300	Lufthansa 6TU climbing FL 300	Correct	ACCU
Lufthansa 6TU climb to FL 300	Lufthansa 6TU FL 300	Correct	ACCU
Lufthansa 6TU climb to FL 300	climbing level 300 Lufthansa 6TU	Correct	ACCU
Lufthansa 6TU climb to FL 300	Lufthansa 6TU climbing FL 330	Wrong	ACCU

Table 4: Data Augmentation of an originally incomplete sample (first row)

3.3 Implementation

The proposed scheme is implemented using the Python programming language in PyCharm Professional IDE. Python was chosen to tap into the open-source neural network library Keras which uses TensorFlow as its backend. Keras was chosen as it offered a simple interface for developing and evaluating neural networks, shifting away from technicalities and towards NN architecture exploration.

Corpus Preprocessing. The collected data was formatted into a CSV file with each sample having four features - instruction, readback, class label and the domain. In order to prepare the data for sentence segmentation, the CSV file was transformed into a pandas data frame and the instructions and readbacks were extracted to build a corpus. It was found that there were inconsistencies in the corpus, the spoken word runway was transcripted as "RWY" sometimes and "runway" other times. In order to resolve inconsistencies, abbreviated words were replaced with the spelt out version. From there, the NLTK library's word_tokenize() function was use to separate individual words within each instruction and readback sentence. Initially, multi-word expressions were learnt and assembled together using the Phraser() from the Gensim library. This way, the sentence "roger direct DIBET Singapore 52 thank you" will be segmented into ['roger', 'direct', 'DIBET', 'Singapore', '52', 'thank_you']. The motivation behind doing this was to allow the CNN to learn patterns in transcribed ATC speech beyond words that naturally occur together in English. However, further along in the research it was decided that the Phraser function should not be used because as a side effect, several call signs were assembled together which other were not i.e. 'CIG_1642' was assembled together while 'CIG 1622' was separated as two tokens. With that, CNN would have difficulties detecting and comparing features because call signs are represented unevenly.As such, the Phraser function was not used because it was critical to keep the processing of call signs standardised. Once the transformations are applied, the instructions and readbacks are separated to output a list of segmented sentences for each.

Text Vectorization. This step transforms the segmented sentence into a sentence vector, with the length of the vector representing the number of words and the width of the vector representing the vocabulary or embedding size. Before vectorization techniques are applied, the text is first padded to a length of 25 which was slightly longer than the length of the longest sentence. Even though previous research found great success in the use of one-hot vectors, the word2vec method was favoured and implemented. Word2vec produces dense vectors of floating-point values in a low dimensional space compared to sparse vectors of binary values in high dimensional space that the one-hot encoding method produces.

Word2vec is a popular method for NLP tasks created by Mikolov (2013) at Google, and has shown promising results in various English NLP tasks (Kim, 2014; Zhang and Wallace, 2017). The CBOW word2vec algorithm predicts a word given its context i.e the words before and after the target word. This embedding technique processes words through a two-layer neural network to create word embeddings that are mapped onto a vector space in a way where the distance between vectors represent their semantic relationship i.e. words that are mapped close together would have similar meaning. Given that the specificity of the corpus influences the applicability of the embeddings on a task more than the size of the corpus (Dusserre and Padró, 2017), a new word2vec model instead of using pre-trained vectors. The Gensim library which is open source on Python was used to create the model, with a min_count =1, to extract every unique word in the dataset, and an embedding size of 32. The dataset used to develop the word2vec model was the corpus made up of both instructions and readback, to maximise the vocabulary of the model. From there, every segmented instruction and readback sentences was encoded into vectors with each word having a length of the embedding size of 32. Thus, the resulting vector for each sentence was of size 25 x 32.

Semantic Extraction. To harness the feature detection capabilities of CNNs, this step uses a bi-stream CNN with two input channels to learn concise semantic vectors

from instructions and readbacks simultaneously. For processing a sentence vector S of size a x b, one-dimensional filters would be applied across the sentence. The CNN would convolve over words in a sentence with windows so the features can be extracted independently of their position within the sentence (Kalchbrenner et al., 2014). A feature f_i can be generated from a window of words $S_{i:i+x-1}$ using Eq (1)

$$f_i = \sigma(W \times S_{i:i+n-1} + bias) \tag{1}$$

where n denotes the size of the window, **W** stands for the weight matrix in convolutional layer and σ represents an activation function. Windows of differing sizes would represent detection of patterns of different sizes, i.e. a window size = 2 would represent learning bigrams while and window size = 5 would represent patterns of 5 adjacent words. Thus, CNNs would be used to learn expressions such as "clear for", "flight level", or "proceed direct to" and identify them in a sentence. By applying the filter to all possible window of words in S, a feature map can be learned by concatenating all the features as seen in Eq (2)

$$f = f_1, f_2, ..., f_{n-h+1} \tag{2}$$

A convolution over the sentence vector produces a feature of size (a - n +1). However, taking inspiration from CNNs used for object recognition in computer vision (Lecun et al., 1998), researchers suggested that multiple feature maps should be applied to enrich the sentence representation and learn complementary features (Kalchbrenner et al., 2014).

In this research, three different CNN architectures will be explored to determine which would contribute to the best instance of the proposed scheme. It should be noted that given the configuration of the bi-stream CNN which features two input channels, it was not possible to use methods such as GridSearch or RandomSearch to optimize the hyperparameters. As such, hyperparameters were tuned manually.

3.3.1 The Cheng & Jia Architecture

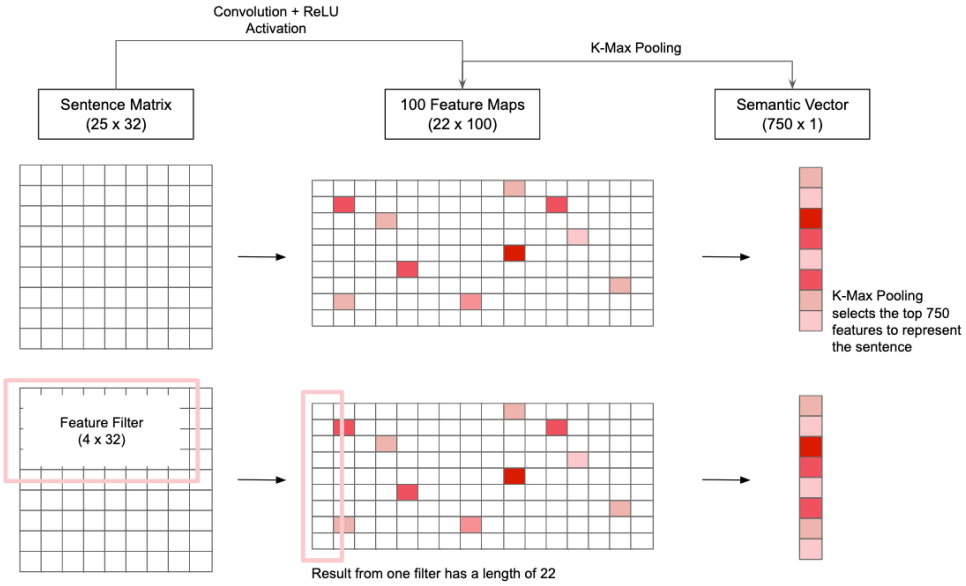


Figure 3: Visualizing the layers in the Cheng & Jia Architecture CNN

This research was inspired by the work of Cheng et al (2018) who achieved impressive results classifying readbacks in Chinese ATC Speech, thus it was only necessary their architecture was replicated. This architecture is a bi-stream CNN featuring two one-layer CNNs composed of an input layer, a convolutional layer and a pooling layer. It takes two zero-padded sentence matrices as an input of size $a \times b$ where a is the sentence length and b is the word vector dimension. Each input is passed through a convolutional filter of size $n \times b$ to generate feature maps. To increase the number of features learnt, multiple filters are applied to generate multiple feature maps. The feature maps are pushed through a k-max-pooling layer to dimensionality reduction and to prevent overfitting. The top k features of all feature maps are extracted and concatenated into a semantic vector with k features as the output.

In terms of hyperparameter tuning, upon training, it was found that the value n of the window size of the convolutional filter was best to set at a value of 4 and the number of feature maps learnt to 100. In the k-max-pooling layer, the value of k should be set to approximately 150 features for optimal performance. For the activation function, ReLU was used for all layers.

3.3.2 The Zhang & Wallace Architecture

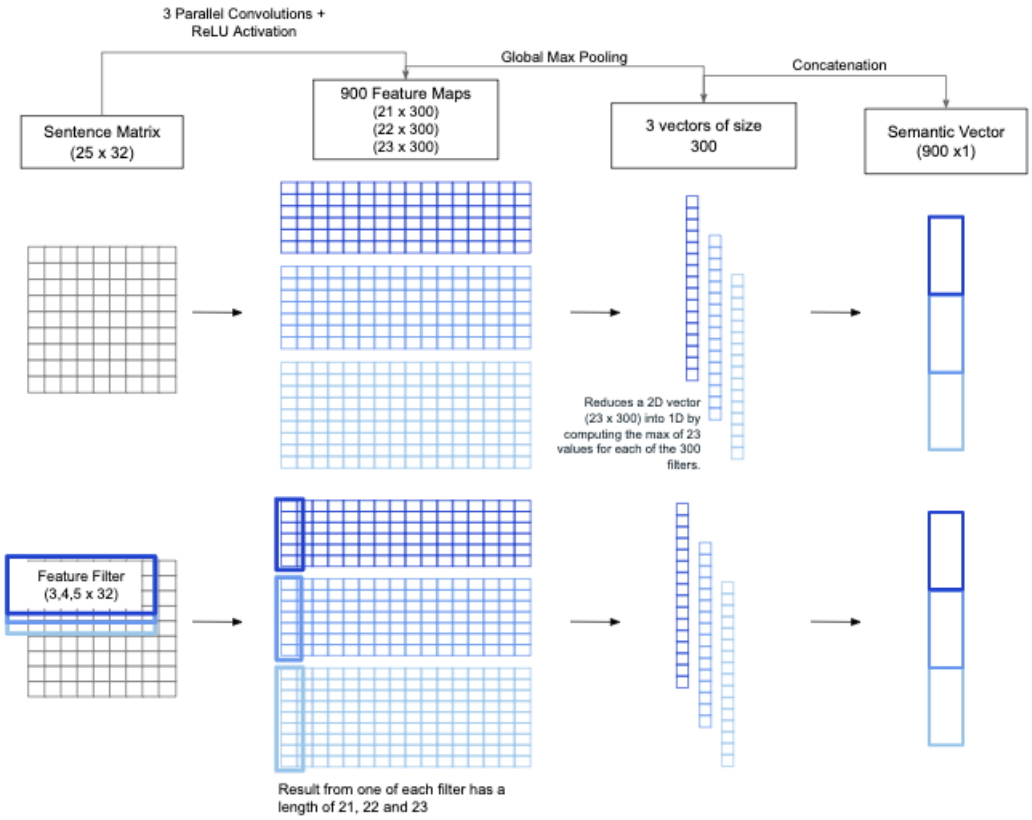


Figure 4: Visualizing the layers in the Zhang & Wallace Architecture CNN

Zhang and Wallace (2017) had done extensive research on using CNN for sentence classification and illustrated a CNN architecture for the task. Their architecture was implemented in such a way to suit our readback classification scheme which features a bi-stream CNN configuration. Each CNN features an input layer, convolutional layers with differing window sizes and a pooling layer. Similar to the Cheng and Jia model, it takes two inputs of zero-padded sentence matrices of size $a \times b$ where a is the sentence length and b is the word vector dimension. However, in this architecture, each input is passed through multiple convolutional filters of sizes $n_{1,2,3} \times b$ simultaneously instead of one. This method is also known as grouping and was introduced in AlexNet (Krizhevsky et al., 2017). The convolutional filters of different sizes allow the network to learn phrases of different lengths and generate feature maps of varying sizes. A one-dimensional global max-pooling layer is then applied on the feature maps produced by the convolutional layers to extract the most prominent feature over the length of the feature map, essentially max pooling over time. This reduces the matrix to a fixed size, the matrices are then concatenated to output a concise semantic vector representing the sentence.

In terms of hyperparameter tuning, in the training phase, it was found that the value $n_{1,2,3}$ of the window sizes of the convolutional filters was best set to 3,4,5, the number of feature maps to 300 and with regards to the activation function, ReLU was used for all layers.

3.3.3 The Novel Architecture

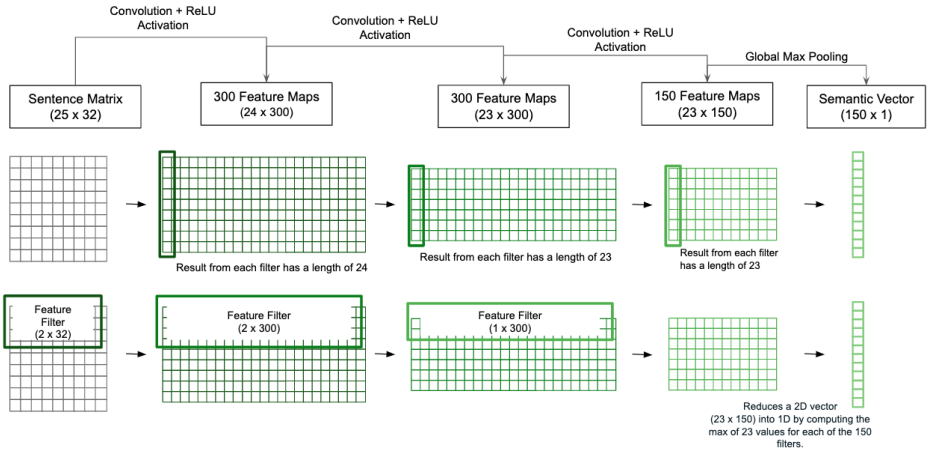


Figure 5: Visualizing the layers in the Novel Architecture CNN

In this research, a novel architecture was designed and implemented to extract semantic value from sentences for this readback classification task. This bi-stream CNN configuration features two relatively deeper CNNs compared to the models discussed above. Additionally, this configuration stacks multiple convolutional layers consecutively instead of alternating convolution and pooling layers as done traditionally. Each CNN has an input layer, three convolutional layers and one pooling layer. The two input channels each take a zero-padded sentence of size $a \times b$, like the aforementioned models, where a is the sentence length and b is the word vector dimension. Each input channel is connected to the first convolutional layer with a filter of size $n_1 \times b$. Instead of passing the feature maps generated from the first layer, they are passed through a second convolutional layer with a filter of size $n_2 \times b$, to extract higher-level semantic features without reducing the matrix dimensions and produce better representation compared to a single convolutional layer (Jeong et al., 2019). The resulting feature maps are then passed through a third consecutive convolutional layer of size $1 \times b$. The third convolutional layer with a kernel size = 1 was incorporated for dimensionality reduction by downsampling the number of feature maps and acts as a channel-wise pooling layer by pooling features across channels (Lin et al., 2014). The feature maps generated from the third convolutional layer are then used as input to a pooling layer to extract the most prominent features of each feature map by applying one-dimensional global max pooling or in other words, max pooling over time.

In terms of hyperparameter tuning, upon training, it was found that the value $n_{1,2}$ of the window sizes of the convolutional filters was best set to 2 for both, the number of feature maps to 300 and for the activation function, ReLU was used for all layers.

Classification. This step matches the semantic vectors of the instructions and readbacks to predict the class label representing the relationship between the I-R pair by merging the bi-stream CNN with an MLP. Mathematically, cosine proximity reflects the angle between two vectors. For classification, cosine proximity - a measure for text similarity (Gunawan et al., 2018; Zahrotun, 2016) between the instruction and readback vectors will be computed and given as part of the input to the MLP. The similarity score for an I-R pair is defined by Eq (3).

$$CosineSimilarity = \frac{I_i^T \times R_i}{\|I_i\| \times \|R_i\|} \tag{3}$$

The input to the MLP is defined as the concatenation of the semantic vector of the instruction, the similarity score, and the semantic vector of the readback. The input is defined by Eq (4).

$$Input_i = [[I_i], [CosineSimilarity_i], [R_i]] \tag{4}$$

The class labels for each I-R pair was predicted by pushing the inputs through an MLP with a visible input, hidden, and dropout and output layers. To prevent the model from overfitting, a dropout layer was incorporated after the hidden layer and a constraint is imposed on the weights of the hidden layer to ensure the max normalization does not exceed a value of 3. The dropout layer was connected to 3 nodes in the output layer, one for each class label, and a softmax activation was applied. With regards to the hyperparameters, it was found that the network performed best with 128 nodes in the hidden layer with a reLU activation and a 30% dropout layer.

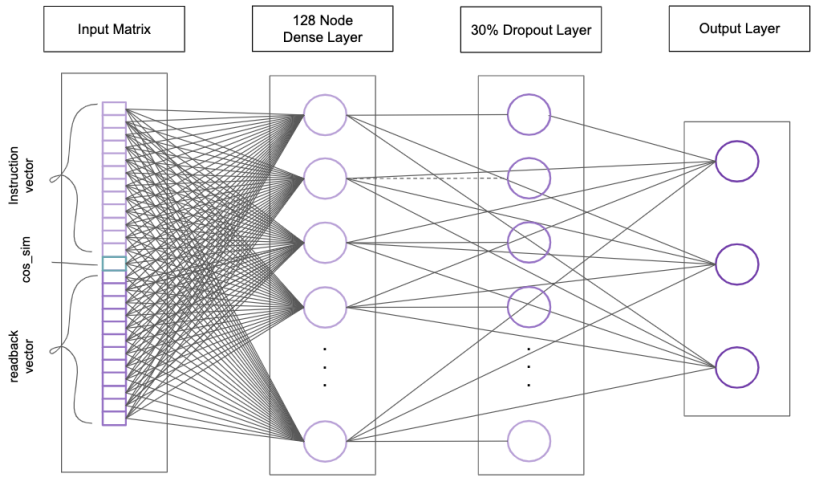


Figure 6: Visualizing the layers in the classifier network

In order to jointly train the bi-stream CNN and the MLP classifier to maximise correct classification of readbacks from pilots based on its semantic similarity to instructions from controllers, they were merged together to form a biCNN-MLP configuration.

When training the biCNN-MLP models, the proportion of training data was increased until the loss reduces over time reasonably well. It was identified that best results were achieved then splitting the dataset into training and test sets with a ratio of 80:20 while keeping the same proportion of classes. Additionally, Keras by default uses the last 10% of the training samples as the validation set to test the model's performance after each epoch. By looking at the validation accuracy and loss values, adjustments can be made to hyperparameters to overcome overfitting or underfitting. As such, the models were trained on 2700 samples, validated on 300 samples after each epoch and tested on 750 samples.

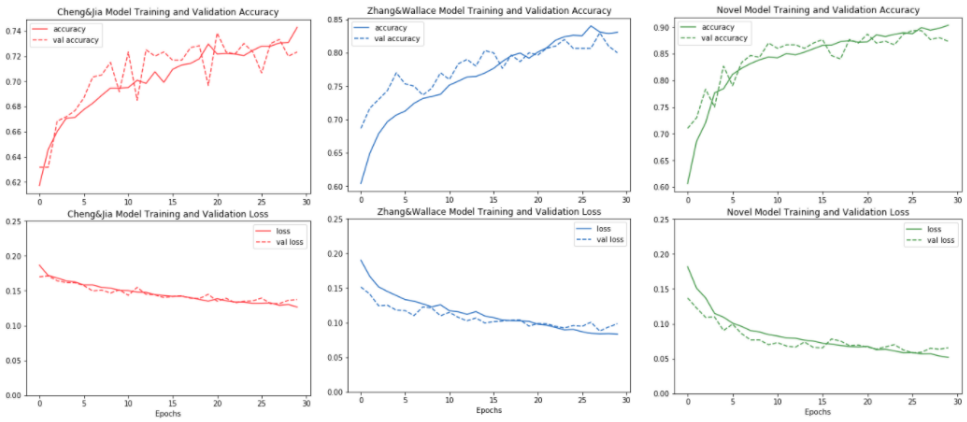


Figure 7: Training and Validation Accuracies and Losses during Implementation depicting that none of the models were underfitting or overfitting

The biCNN-MLP models were trained using the Adaptive Moment Estimation (Adam) optimizer for 30 epochs with a batch size of 16. Categorical cross-entropy was used as the loss function and the metric used was accuracy. As mentioned before, it was necessary to tune these models manually as they feature two input streams. However, the training and validation accuracy and loss values were analysed to ensure that the models did not suffer from underfitting or overfitting as seen in the graphs above. It was observed that with these training configurations, all three models were trained adequately since the training losses were approximate to the validation losses. Additionally, across all models, the graphs of the training and validation accuracy values hover around a small range of values towards the last few epochs.

4 Experiment and Results

This section is aimed at addressing two questions in this research - which implementation of the proposed scheme should be adopted and 2) with this implementation, it is possible to automate the readback classification?

The metrics for comparing the performance across the models and the metrics for determining whether the proposed scheme can be recommended for hearback automation will be defined. With that, experiments to address the two questions will be conducted.

4.1 Performance Evaluation Metric and Baseline

Given there are three models implemented from the proposed scheme each with a different CNN architecture, the performances of these models will be compared. With regards to evaluation metrics, while accuracy gives a general idea of how the model is performing, F-scores provide a more comprehensive understanding of the model performance. Accuracy is the proportion of true positives and true negatives across all the samples and is mathematically defined by Eq (5).

$$Accuracy = \frac{TruePositives + TrueNegatives}{AllSamples} \tag{5}$$

On the other hand, F-score is the weighted harmonic mean of precision and recall for each

class in case of multi-class classification. In order to compare the f-scores across models, the per-class f-scores have to be combined into an overall f-score. The overall F-score for the model is taken by calculating the macro-average of the per-class f-scores. The formula for calculating overall f-score, F-score, precision and recall is mathematically defined by Eq (6), (7), (8) and (9) respectively.

$$OverallF - score = \frac{F_1 + F_2 + F_3}{3} \tag{6}$$

$$F - score = 2 \cdot \frac{Precision \times Recall}{Precision + Recall} \tag{7}$$

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{8}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{9}$$

The models will be compared against a baseline rate of accuracy to determine whether the models were capable of learning patterns in the dataset. The baseline for accuracy is derived by selecting the majority class and using it as the prediction for all samples, and is taken as 60%. Additionally, to identify the best implementation for the proposed scheme, a comparison will be made between the three models, the performance of each model will be evaluated based on its accuracy and overall f-score for each class label.

With regards to addressing the research question - whether it is possible to develop an effective readback classifier capable of automating the hearback process, it is necessary to compare the performance of the model against that of a human controller. The controller's recall rate corresponds to the proportion of readback errors they correct when making hearbacks, the most recent hearback recall rate was reported to be at 77.8% (Lennertz, 2017). As such, it is critical to look at the proportion of samples from the minority class correctly labelled. The recall metric is used to evaluate whether automation of the hearback process is possible. It is especially critical to compare recall rates for readbacks from the classes "Incomplete" or "Wrong", as this can be directly measured against the controller's recall rate. The proposed scheme and its best implementation can be recommended to automate the hearback process if the recall rate of the model outperforms that of the human controllers.

4.2 Performance Comparison

In order to obtain experimental results, repeated random subsampling i.e. repeated Monte Carlo cross-validation or repeated holdout, was done 15 times on each model. The I-R pairs in each run were randomly divided into fixed-sized training and test sets. This strategy allows for more possible combinations of samples in each set and can be repeated numerous times without compromising the size of the training and test set. For example, with 3750 samples, the model can be trained on 3000 and evaluated on 750 samples for an arbitrary number of times but this is limited to k=5 if k-fold cross-validation is used. Repeated random subsampling was selected because it ensured randomness to a great extent and offers flexibility in determining the sizes of training and test sets.

First, 15 random numbers between 0-99 would be generated. This list of random numbers served as random states to be given as input when calling the train_test_split() function, which would result in 15 training and test splits. This was done to ensure

consistent testing across the models and replicability of the experiments. Then, for each split the model would be trained on the training set and afterwards evaluated on the test set. The accuracy and loss of each test would be reported, and a list of predicted values y_pred would be obtained by passing the test split to the predict() function. The list y_pred would then be compared against a list of target values y_true, to create a confusion matrix. From the confusion matrix, metrics such as recall, precision and f-score can be calculated and averaged over the 15 splits.

	Model Accuracy	**Loss**	**Mean Overall F-score**	**'Correct' Mean F-score**	**'Incomplete' Mean F-score**	**'Wrong' Mean F-score**
Cheng & Jia	71.8	0.13	54.7	82.7	68.8	12.7
Zhang & Wallace	78.5	0.11	70.3	86.3	73.5	51.0
Novel	84.8	0.09	79.4	88.9	80.3	68.9

Table 5: Summary Statistics of Model Performances given in percentages (%)

Findings reveal that in terms of accuracy all the models outperform the baseline of 60%, even the Cheng & Jia Model which performed the worst had a mean accuracy of 71.8%. The Zhang & Wallace Model performed better than the Cheng & Jia Model with a mean accuracy of 78.6% however, the spread of its accuracy values is the biggest. On the other hand, the Novel Model outperformed the baseline as well as both other models, with the highest mean accuracy of 84.8%. The Novel Model reached above 86% accuracy in some training and test splits. Moreover, its accuracy values do not vary as much as the other models as observed in the small vertical spread in its box plot. With regards to f-scores, the same patterns were observed. The Cheng & Jia Model performed worst with an overall f-score of 54.7% averaged over the 15 splits, the Zhang & Wallace Model had an average overall f-score of 70.3% and the Novel Model had the highest average overall f-score with a value of 79.4%.

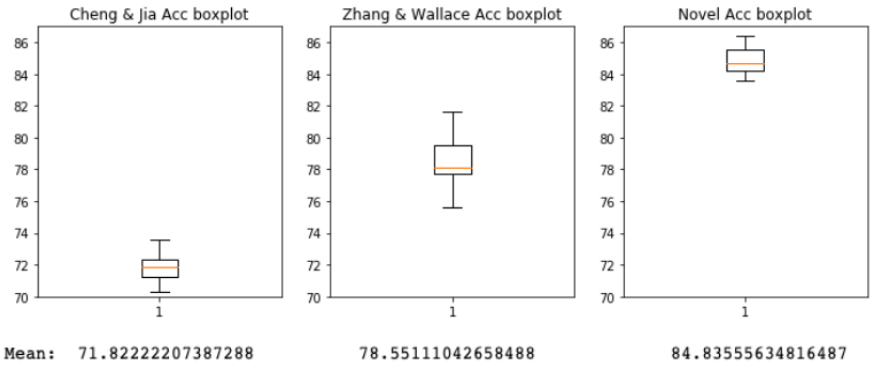


Figure 8: Boxplots representing the accuracies of each model on the test dataset

4.3 Proposed Scheme for Hearback Process Automation

Precision and recall rates were also recorded during each experiment. Findings show that across all models, recall rates for the "Correct" class was always the highest, followed by

Label	Cheng & Jia		Zhang & Wallace		Novel	
	Precision	Recall	Precision	Recall	Precision	Recall
Correct	73.9	93.9	80.7	92.9	86.3	91.7
Incomplete	69.8	68.6	80.8	68.0	80.4	80.7
Wrong	45.5	7.8	64.0	43.7	78.0	62.1
All Samples	63.1	56.8	75.2	68.2	81.6	78.2

Table 6: Precision and Recall Performance of Models given in percentages (%)

	Comparable Recall Rate		
Human	77.8 (2017)	8.0 (2006)	60.0 (1997)
Cheng & Jia	51.8		
Zhang & Wallace	69.9		
Novel	78.2		

Table 7: Precision and Recall Performance of Models given in percentages (%)

"Incomplete" and the rates were lowest for the I-R pairs in the "Wrong" class. This is reflected in the average per-class recall rates. Similar to accuracy and f-score, the Cheng & Jia model performed the worst and the Novel model performed the best in terms of precision and recall as well.

However, to compare the recall rates of the model to the controller's recall rates, it was necessary to convert the classification from a multiclass problem to a binary problem. This was because the recall rate reported by the researchers (Lennertz, 2017) reflected that out of 9 readback errors 7 were detected, i.e. the readbacks were classified into two categories - correct and erroneous. Therefore, to identify how many erroneous samples were detected out of all erroneous samples i.e. the comparable recall rate, for each test a second confusion matrix is generated. This second confusion matrix is generated using the multilabel_confusion_matrix() function from the Sklearn library, it is binarized with the "Correct" as the positive class while "Incomplete" and "Wrong" class labels were combined to be the negative or "Erroneous" class. With that, the comparable recall rate for the model is calculated by averaging the recall rates for the "Erroneous" class over 15 tests. The "Erroneous" recall rates from the binarized classification is tabulated below. Similar to the multilabel classification, the Novel model performed best, followed by the Zhang & Wallace model and the Cheng & Jia model performed worst. When compared to the human recall rates, only the Novel model outperformed all three statistics of the human controllers across three different years.

5 Discussion

In this section, the findings will be discussed with regards to the goals of the experiments i.e. to determine which model to be used for the proposed scheme as well as to answer the research question.

5.1 Best Implementation for Proposed Scheme

Findings suggest that the Novel model should be adopted when implementing The Proposed Scheme, it performed best in comparison to the Cheng & Jia and Zhang & Wallace models. Beyond being able to learn patterns correctly from the training samples, the Novel model had the highest f-scores and accuracy. This shows that the model is capable of performing well on imbalanced datasets as f-scores take false positives and false negatives into consideration. Upon further inspection, it was observed that the Novel model had good f-scores across all class labels while the other models did not and suffered especially in classifying samples from the 'Wrong' class. The Novel model accurately classified many 'Wrong' samples (i.e. good recall) and did not classify many 'Incomplete' or 'Correct' readbacks as 'Wrong' (i.e. good precision). This suggests that the Novel model was the only model capable of distinguishing wrong samples from incomplete or correct ones.

However, in contrast with previous research (Cheng et al., 2018), the Cheng & Jia model performed poorly on the Readback Dataset developed in this research. Unlike its performance on the CRCC data with 90.2% accuracy (without doubling the sentences), it had a mean accuracy of 71.8% in this research on the Readback Dataset. This model was not able to learn the patterns in the Readback Dataset well, despite a simpler classification problem with 3 class labels in this research as compared to the 6 class labels which Cheng et al had. Additionally, the model was trained with a significantly higher number of trainable parameters in this research in an attempt to capture more complicated functions i.e. 100 feature maps were learnt and k was set to 750 as compared to 50 and 7 respectively set by Cheng et al. To abolish the difference in word embedding technique resulting in a significant difference in model performance, One-Hot Vectorization was applied on the Readback Dataset. Training took much longer, and the test accuracy was approximately the same. Moreover, the model was heavily overfitted and could not capture any relevant patterns as validation accuracy did not see any steady increase even after 30 epochs.

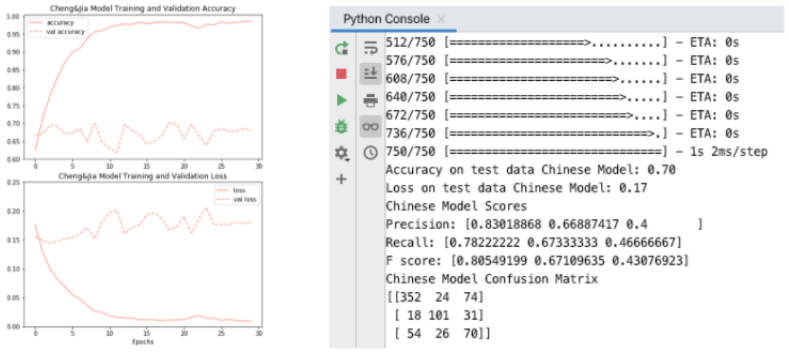


Figure 9: Training & Validation Graph showing overfitting (left) and screenshot of PyCharm console showing performance of model (right)

A plausible reason for the difference in performance could be linguistic differences between Mandarin and English. Mandarin is an uninflected language i.e. tenses do not exist, and the language does not feature phrasal verbs such as "take on" or "up to". Moreover, Chinese words are logograms where each symbol represents a meaning and not a sound, this is very different compared to English especially in the context of transcribed ATC speech. The example I-R pair from the CRCC dataset was concise and exact. In each I-R pair, every phrase in the readback can be matched to a phrase in the instruction, i.e. the exact same phrasing was used. Additionally, sentences did not contain any noise such as greetings or reordering of elements. On the other hand, in English ATC phrases such as "flight level" were used interchangeably with "level" and sometimes omitted altogether and only the values repeated. Nuances in English ATC speech would make it more difficult for the models to identify relevant patterns for readback classification.

Delving into the details of the models, findings suggest that global max pooling performs better than top-k pooling as the Zhang & Wallace and the Novel models significantly outperformed the Cheng & Jia model. This is in line with previous work (Zhang and Wallace, 2017), suggesting that it is necessary to keep n-grams intact but its position within the sentence is not relevant. Also congruent with previous findings which suggest that shallow CNNs were capable of capturing the semantics of a sentence, all three CNN models were capable of performing above a baseline threshold of 60%. However, the best performing model featured three consecutive convolutional layers, outperforming two other models which applied only one convolution to word embeddings. With that, this suggests that CNN configurations for NLP should not be limited to only one-layer convolutions that most researchers propose (Cheng et al., 2018; Kim, 2014; Zhang and Wallace, 2017). There are advantages to learning abstract higher-level features, and channel-wise pooling in NLP by applying more than one convolutional layer consecutively.

5.2 Hearback Automation

Besides performing better than the other models, findings reveal that the Novel model had higher recall rates than that of controllers when differentiating between correct and erroneous I-R pairs. High recall rates represent a low amount of false negatives, suggesting that the model is able to distinguish between the correct and erroneous classes well. The recall rate of the Novel model of 78.2% was higher than the most recent controller hearback recall rate which was reported to be 77.8% as researchers found 7 out of 9 readback errors were corrected (Lennertz, 2017). Additionally, the Novel model outperformed controller recall rates reported in older research as well (Burki-Cohen, 1995; Cardosi and Han, 1997; Prinzo et al., 2006). Given that the Novel model outperforms human controllers, the experimental results show that the proposed scheme is effective in facilitating the automation of the hearback process by classifying readback errors when the Novel model is implemented.

5.3 Relevance of Results

By conducting research on the applicability of CNNs and NLP techniques on hearback process automation, this study contributes to existing literature particularly with regards to sentence matching. This would contribute to the increasing research of CNNs on textual data through a new CNN configuration to extract semantic features. Moreover, this research could be an introduction to addressing hearback automation in English ATC. On the other hand, beyond scientific knowledge, this study also has social

relevance. With The Proposed Scheme and the Novel model, the hearback process could be automated or computer-aided so as to reduce the likelihood of aviation incidents and miscommunication in ATC through the decrease in mental load of controllers as well as the rate of miscommunication in ATC.

6 Conclusion

In this section, the research as a whole will be discussed with regards to the research question. This would include limitations and suggestions for future work as well.

6.1 The Research Question

In conclusion, the results of this research gave insight as to how the hearback process could potentially be automated. This research builds on previous research investigating the use of neural networks, specifically CNNs to extract semantic features in textual data to perform sentence classification tasks. Specifically for hearback automation, the results of current research has empirically shown that CNNs are capable of classifying readbacks into three categories - Correct, Incomplete and Wrong relatively well in comparison to human controllers. The Proposed Scheme developed in this research together with its recommended implementation, the Novel model, was able to identify readback errors better than controllers.

6.2 Research Limitations

However, it is necessary to realize the limitations of this research. The proposed scheme is not standalone, it depends on language technologies capable of perceiving and transcribing ATC speech accurately. In this research, the transcriptions were compiled from only one source, the ATCC corpus. This was due to scarcity of publicly available ATC speech, much less those which are transcribed. While the ATCC contains communications from three different domains and a few airspaces, the data contained spelling errors and incomplete transcriptions. As such, the amount of I-R pairs that could be extracted were limited.

On top of that, it was also difficult to find available existing research analysing controller-pilot communications from recent years. This could be because they may not have been made publicly available or fewer analysis had been conducted. In the 1990s, there was a series of reports analysing communication across different domains. However in 2006 and 2017 there was only one, making it challenging to generalize statistics reported in these reports or use these statistics to make meaningful comparisons. Looking at the most recent report, in 2017 researchers analysed a set of 1,169 instructions and clearances from ten hours of communication from the Kansas City Air Route Traffic Control Centre. It was reported that from the ten hours of communication, 9 readback errors were identified making up $\leq 1\%$ of all readbacks (Lin et al., 2014). Additionally, researchers, in the analysis conducted in 2006, had classified readbacks differently from controllers. It appeared that controllers evaluated the intrinsic safety component of each readback before deciding whether or not to correct a detected error while researchers followed the readback requirement guidelines strictly. In other words, it is difficult to find a reliable statistic about controllers' hearback recall and as such it is challenging to make definitive comparisons.

6.3 Future Work

For further improvements, the Readback Dataset should compile data across several sources and the quality of transcriptions from those sources should be verified. Moreover, utilizing a recent controller hearback recall rate with high construct validity by ensuring that both researchers and controllers follow the same guidelines when classifying readbacks and using a large enough dataset could also be recommended. This should hold true for both analysing human recall rates as well as model recall rates. With the use of a valid dataset and a reliable statistic for controller hearback recall rate, the predictive validity of the research would be heightened.

On the other hand, given the performance of the Novel model on the Readback Dataset, it may be interesting to improve it by combining rule-based methods with CNNs. This would allow CNN to learn more complex patterns that cannot be captured by the ruled-based methods. Alternatively, it would also be useful to investigate how the interaction between the instruction and readback sentences in the sentence representation formation process affect the performance of the model. Additionally, beyond ATC, future research could look into the applicability of CNN and feature extraction on other air services such as filtering of NOTAMs.

References

Agarwal, B., Ramampiaro, H., Langseth, H., and Ruocco, M. (2018). A Deep Network Model for Paraphrase Detection in Short Text Messages. *Information Processing & Management*, 54(6):922–937.

Billings, C. E. and Cheaney, E. S. (1981). *Information Transfer Problems in the Aviation System*. National Aeronautics and Space Administration (NASA) Scientific and Technical Information Branch, Washington D.C., USA.

Breul, C. (2013). Language in Aviation: The Relevance of Linguistics and Relevance Theory. *LSP Journal - Language for Special Purposes, Professional Communication, Knowledge Management and Cognition*, 4(1):70–87.

Burki-Cohen, J. (1995). An Analysis of Tower (Ground) Controller - Pilot Voice Communications. Technical report, John A. Volpe National Transportation Systems Center, Cambridge, Massachusetts, USA.

Cardosi, K., Falzarano, P., and Han, S. (1998). Pilot-Controller Communication Errors: An Analysis of Aviation Safety Reporting System (ASRS) Reports. Technical report, John A. Volpe National Transportation Systems Center, Cambridge, Massachusetts, USA.

Cardosi, K. M. and Han, S. (1997). An Analysis of TRACON (Terminal Radar Approach Control) Controller- Pilot. Technical report, John A. Volpe National Transportation Systems Center, Cambridge, Massachusetts, USA.

Chen, S., Kopald, H. D., Chong, R., Wei, Y., and Levonian, Z. (2017). Read Back Error Detection using Automatic Speech Recognition. In *12th USA/Europe Air Traffic Management Research and Development Seminar (ATM2017)*, Seattle, Washington, USA. June 26-30, 2017. DOI: 10.1109/AICCSA.2016.7945669.

Cheng, F., Jia, G., Yang, J., and Li, D. (2018). Readback Error Classification of Radiotelephony Communication Based on Convolutional Neural Network. In *Chinese Conference on Biometric Recognition*, pages 580–588, Urumqi, China. Springer International Publishing. August 11-12, 2018. DOI: 10.1007/978-3-319-97909-062.

Civil Aviation Authority (2015). *CAP 493: Manual of Air Traffic Services Part 1*. TSO (The Stationery Office), West Sussex, UK.

Civil Aviation Authority (2016). *CAP 413 Radiotelephony Manual*. TSO (The Stationery Office), West Sussex, UK, 22 edition.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Dusserre, E. and Padró, M. (2017). Bigger does not Mean Better! We Prefer Specificity. In *12th International Conference on Computational Semantics — Short papers*, Montpellier, France. September 19-22, 2017.

Es, G. v. (2004). Air-ground communication safety study: an analysis of pilot-controller occurrences. Technical report, European Organisation for the Safety of Air Navigation (EUROCONTROL), Brussels, Belgium.

Gontar, P., Schneider, S. A. E., Schmidt-Moll, C., Bollin, C., and Bengler, K. (2017). Hate to Interrupt You, but... Analyzing Turn-arounds From a Cockpit Perspective. *Cognition, Technology & Work*, 19(4):837–853.

Gunawan, D., Sembiring, C., and Budiman, M. (2018). The Implementation of Cosine Similarity to Calculate Text Relevance between Two Documents. *Journal of Physics: Conference Series*, 978(1).

Hu, B., Lu, Z., Li, H., and Chen, Q. (2014). Convolutional Neural Network Architectures for Matching Natural Language Sentences. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 2042–2050, Montreal, Canada. December 8-13, 2014. DOI:10.5555/2969033.2969055.

International Air Transport Association (2018). IATA Forecast Predicts 8.2 billion Air Travelers in 2037. Online; posted 24 October 2018 at https://www.iata.org/en/pressroom/pr/2018-10-24-02/.

International Civil Aviation Organization (1994). *Annex 13 to the Convention on International Civil Aviation*. ICAO.

International Civil Aviation Organization (2001). *Annex 10 to the Convention on International Civil Aviation: Aeronautical Telecommunications*. ICAO.

Islam, A. and Inkpen, D. (2008). Semantic Text Similarity using Corpus-Based Word Similarity and String Similarity. *ACM Transactions on Knowledge Discovery from Data*, 2(2):1–25.

Jeong, Y.-S., Woo, J., and Kang, A. R. (2019). Malware Detection on Byte Streams of PDF Files Using Convolutional Neural Networks. *Security and Communication Networks*, 2019:1–9.

Jia, G., Cheng, F., Yang, J., and Li, D. (2018). Intelligent Checking Model of Chinese Radiotelephony Read-Backs in Civil Aviation Air Traffic Control. *Chinese Journal of Aeronautics*, 31(12):2280–2289.

Jia, G., Lu, Y., Lu, W., Shi, Y., and Yang, J. (2017). Verification Method for Chinese Aviation Radiotelephony Readbacks Based on LSTM-RNN. *Electronics Letters*, 53(6):401–403.

Johnson, R. and Zhang, T. (2015). Semi-supervised Convolutional Neural Networks for Text Categorization via Region Embedding. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, pages 919–927, Montreal, Canada. December 7-12, 2015.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A Convolutional Neural Network for Modelling Sentences. In *The 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 655–665, Maryland, USA. June 22-27, 2014. DOI:10.3115/v1/P14-1062.

Kenter, T. and de Rijke, M. (2015). Short Text Similarity with Word Embeddings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management - CIKM '15*, pages 1411–142, Melbourne, Australia. Oct 19-23, 2015. DOI:10.1145/2806416.2806475.

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. October 25-29, 2014. DOI: 10.3115/v1/D14-1181.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the Association for Computing Machinery*, 60(6):84–90.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lennertz, T. (2017). Analysis of Controller-Pilot Voice Communications from Kansas City Air Route Traffic Control Center. Technical report, John A. Volpe National Transportation Systems Center, Washington, D.C.

Lin, M., Chen, Q., and Yan, S. (2014). Network In Network. In *2nd International Conference on Learning Representations, ICLR 2014, Conference Track Proceedings*, pages 1–10, Banff, AB, Canada. Apr 14 - 16, 2014.

Mihalcea, R., Corley, C., and Strapparava, C. (2006). Corpus-based and Knowledge-based Measures of Text Semantic Similarity. In *AAAI'06: Proceedings of the 21st national conference on Artificial intelligence - Volume 6*, pages 775–780, Boston, Massachusetts, USA. July 16–20, 2006. DOI:10.5555/1597538.1597662.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. In *Conference: Proceedings of the International Conference on Learning Representations (ICLR 2013)*, Arizona, USA. May 2-4, 2013.

Patty, A. (2016). Fatal Consequences of Miscommunication between Pilots and Air Traffic Controllers. *The Sydney Morning Herald*. October 2, 2016.

Prinzo, O. V., Hendrix, A. M., and Hendrix, R. (2006). The Outcome of ATC Message Complexity on Pilot Readback Performance. Technical report, Federal Aviation Administration, Oklahoma, USA.

Schroeder, D., Bailey, L., Pounds, J., and Manning, C. (2007). A Human Factors Review of the Operational Error Literature. Technical report, Federal Aviation Administration, Oklahoma, USA.

Severyn, A. and Moschitti, A. (2015). Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382, Santiago, Chile. Association for Computing Machinery. August 9-13, 2015. DOI:10.1145/2766462.2767738.

Simard, P., Steinkraus, D., and Platt, J. (2003). Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)*, pages 958–962, Edinburgh, Scotland. August 3-6, 2003, DOI: 10.1109/ICDAR.2003.1227801.

Smídl, L., Švec, J., Tihelka, D., Matoušek, J., Romportl, J., and Ircing, P. (2019). Air traffic control communication (ATCC) speech corpora and their use for ASR and TTS development. *Language Resources and Evaluation*, 53(3):449–464.

Xing, J. and Bailey, L. L. (2005). Attention and memory in air traffic control tasks. *Journal of Vision*, 5(8):427–427. Publisher: The Association for Research in Vision and Ophthalmology.

Yin, W. and Schütze, H. (2015). Convolutional Neural Network for Paraphrase Identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911, Denver, Colorado. Association for Computational Linguistics. May 31, 2015 - Jun 5, 2015. DOI: 10.3115/v1/N15-109.

Zahrotun, L. (2016). Comparison Jaccard similarity, Cosine Similarity and Combined Both of the Data Clustering With Shared Nearest Neighbor Method. *Computer Engineering and Applications Journal*, 5(1):11–18.

Zhang, Y. and Wallace, B. (2017). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. In *Proceedings of the The 8th International Joint Conference on Natural Language Processing*, page 253–263, Taipei, Taiwan. Nov 27, 2017 - Dec 1, 2017.

A The Readback Dataset

The Readback Dataset is made up of instruction and readback (I-R) pairs extracted from the ATCC corpus published by Šmídl et al (2019) from University of West Bohemia, Department of Cybernetics for the research project "Intelligent technologies for improving air traffic security (IT-BLP)". 750 I-R pairs in total, 250 from each domain, were extracted randomly from the TRS files published. To ensure the pairs were randomly extracted, a short AppleScript program was used to assign 3-digit number to the beginning of each filename. When entering this script into AppleSript, a prompt will ask for a folder to be chosen. All files in the chosen folder will be have a 3-digit random number prefix attached to the filename, which is useful for randomly sorting the files. The script is as follows:

```
set a to choose folder
tell application "Finder"
    set all_Files to every file in folder a
    repeat with a_file in all_Files
    set a_file's name to ((random number from 100 to 999) as text) & a_file's name
    end repeat
end tell
```

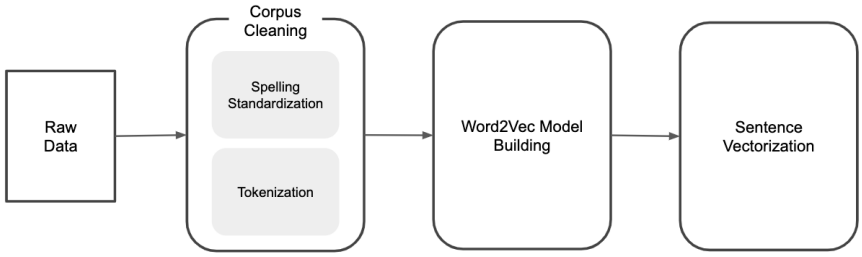
Each pair was augmented in such a way that that there will be 3 correct, 1 incomplete and 1 wrong readbacks for each unique instruction. With that, there were 3750 samples in total after augmentation. There were 1,250 samples from each domain with 750 being correct, 250 incomplete and 250 wrong. All the samples were then shuffled resulting in The Readback Dataset.

A portion of The Readback Dataset is shown in the following page. The dataframe is structure with 4 columns 'Instruction', 'Readback', 'Label' and 'Domain'. The full dataframe has 3750 rows corresponding to each sample.

Table 8: The Readback Dataset extract

Instruction	**Readback**	**Label**	**Domain**
CSA 788 contact Bratislava 125.965	contacting 25.965 CSA 788	Correct	ACCU
Speedbird 856 runway 13 clear to land wind 030 5 knots	Speedbird 856 clear to land runway 13	Correct	TWR
Lufthansa 1YH descend to FL 290 contact Praha 127.125 good bye	Lufthansa 1YH level 290 Praha 127.125 Lufthansa 1YH	Correct	ACCU
CSA 4BE RWY 31 cleared for take off wind 190 degrees 5 knots	CSA 4BE take off RWY 31	Correct	TWR
CSA 2DZ radar radar contact climb to FL 160	CSA 2DZ climbing to FL 160	Correct	APP
Lot 525 to the left to ground 121.9	left to ground 121.9 Lot 525 thank you	Correct	TWR
Air France 1782 descend FL 60	down 60 Air France 782	Wrong	APP
CSA 978 contact Bratislava 134.475	Bratislava radar 34.475 CSA 978	Correct	ACCU
Swiss 498C line up RWY 13	line up 13 Swiss 499C	Wrong	TWR
French Navy 55A2 wind 190 degrees 5 knots RWY 31 clear to land	clear to land 31 French Navy 55A2	Correct	TWR
Ryanair 5JU climb to flight level 340	flight level 340 Ryanair 5UJ	Wrong	ACCU
Speedbird 4AC runway 31 clear to land wind 180 degrees 3 knots	Speedbird 4AC clear to land on runway 31	Correct	TWR
Malaysian 21 hello radar contact maintain FL 330	level 330 Malaysian	Incomplete	ACCU
Swedestar 051 contact Ruzyně tower 118.1 good bye	Swedestar 051	Incomplete	APP
Lufthansa 6RP contact Praha 135.135 good bye	Lufthansa 6RP	Incomplete	ACCU
Lot 354 descend FL 310	FL 310 Lot 354	Correct	ACCU
Lufthansa 2EL descend FL 300 to reach by RAPET	descend level 300 to reach by RAPET Lufthansa	Incomplete	ACCU
Austrian 703 reduce speed to 170 knots to 4 miles final contact tower 118.1	reducing speed Austrian 703	Incomplete	APP
Air Malta 539 radar departure terminated proceed direct to CERNO	Air Malta 539 direct CERNO thank you	Correct	APP
Skytravel 1011 RWY 13 cleared to land wind 020 degrees 3 knots	cleared to land 13 Skytravel 1010	Wrong	TWR
CSA 3CZ radar radar contact descend FL 100 high speed approved and O	descending FL 100 high speed approved check O CSA 3CZ	Correct	APP
Shamrock 661 contact Praha 135.460 good bye	135.460 Shamrock	Incomplete	ACCU
CSA 1ZA RWY 31 cleared for immediate take off wind 170 degrees 4 knots	i have 31 clear for take off CSA 1ZA	Correct	TWR
Wizzair 491 runway 31 clear for take off wind 100 degrees 3 knots	clear take off runway 31 Wizzair 491 bye bye	Correct	TWR
Lot 391 proceed to LETKO	Lot 391 LETKO directly	Correct	ACCU
Lufthansa 4UA climb FL 340	climb FL 340 Lufthansa 4UA	Correct	ACCU
CSA 406 RWY 13 clear for take off wind 170 degrees 1 knots	CSA 406 cleared for take off RWY 13	Correct	TWR
Lufthansa 4UA contact Rhein 133.285	contact 133.285	Incomplete	ACCU
Bline 70V contact radar 120.275 good	okay contact 120.275 Bline 70V	Correct	APP
CSA 838W descend FL 90	descend level 90 CSA 838W	Correct	APP
Easy 5487 Praha hello radar contact descend to FL 290 fifteen hundred or more	descend FL 290 at fifteen hundred or more Easy 5487	Correct	ACCU
CSA 635 reduce speed 250 knots or less	speed 230 or less CSA 635	Wrong	APP
CSA 508 proceed to VENOX	proceed to VENOX CSA 508	Correct	APP
CSA 573 proceed to ARVEK i'm sorry for late call	direct to ARVEK CSA 73	Wrong	APP
CSA 978 proceed to MAKAL	proceed to MAKAL CSA 978	Correct	APP
Air France 132 reduce speed 240 knots or less	redcing speed 240 or less Air France 132	Correct	APP
Austrian 336N Praha radar contact descend FL 290 1 minimum	descending 290 with 1 thousand feet per minute Austrian 336N	Correct	ACCU
Easy 247F for climb contact Praha 133.410 good bye	Contact 133.410 Easy 247F	Correct	ACCU
Donavia 306 climb to FL 160	continue climb FL 160 Donavia 06	Wrong	APP

B Pre-Processing Pipeline



In this research the corpus is made up of the items in the 'Instruction' and 'Readback' columns of the dataframe. The goal is the preprocessing module is to transform the sentences into a format that can be manipulated by the CNNs in the next step. The preprocessing can be broken down into corpus cleaning and word embedding. Corpus Cleaning is handled by the 'dataset_builder.py' file and the word embedding is handled by the 'w2v_embedder.py' file.

These are the functions of the 'dataset_builder.py' file: _init_(), build_data(), get_corpus(), get_instruction(), get_readback(), get_label(). The main function in this file is the build_data() function while the others were merely utility getter functions. The build_data() function is defined as follows:

```
def build_data(self):
    df = pd.read_csv('dataset.csv')
    instruction = df['Instruction'].values.tolist()
    readback = df['Readback'].values.tolist()
    label = df['Label'].values.tolist()
    corpus = instruction + readback
    corpus = [word.replace('RWY', 'runway') for word in corpus]
    corpus = [word.replace('FL', 'flight_level') for word in corpus]
    tok_corp = [nltk.word_tokenize(text) for text in corpus]
    slice = int(len(tok_corp)/2)
    return tok_corp, tok_corp[:slice], tok_corp[slice:], label

def __init__(self, tok_corp):
        self.w2v_data = tok_corp
        self.atc_model = gensim.models.Word2Vec(self.w2v_data,
                            min_count = 1, size = 32)
        self.max_len = 25

def vectorize(self, data):
    vectors = [self.vectorize_sen(sentence) for sentence in data]
    return vectors

def vectorize_sen(self, sentence):
    sentence_vec = np.zeros((self.max_len, self.atc_model.vector_size))
    for i,word in enumerate(sentence):
         sentence_vec[i] = self.atc_model[word]
    return sentence_vec
```

C CNN Implementations

The Python code for each implementation is given under the corresponding subsections. The code for evaluating each model was the same, and is shared uner the subsection 'Model Evaluation'.

C.1 The Cheng & Jia Architecture

```
def build_model(self):
    print("\n—— Create Cheng & Jia network ———\n")
    instruction_input = Input(shape=(self.num_words, self.embedding_size,)
                        ,dtype='float32', name='instruction_input')
    instruction = Conv1D(100, 4, activation='relu', input_shape=(self.num_words,
                    self.embedding_size))(instruction_input)
    instruction = Flatten()(instruction)
    instruction = Lambda(lambda x: tf.nn.top_k(x, k=150, sorted=True).values)(instruction)

    readback_input = Input(shape=(self.num_words, self.embedding_size,), dtype='float32',
                    name='readback_input')
    readback = Conv1D(100, 4, activation='relu', input_shape=(self.num_words,
                self.embedding_size))(readback_input)
    readback = Flatten()(readback)
    readback = Lambda(lambda x: tf.nn.top_k(x, k=150, sorted=True).values)(readback)

    cos_sim = dot([instruction, readback], axes=1, normalize=True)
    final = concatenate([instruction, cos_sim, readback])
    final = Dense(units=128, activation="relu", kernel_constraint=max_norm(3))(final)
    final = Dropout(0.3)(final)
    output = Dense(self.num_classes, activation='softmax')(final)
    cj_model = Model(inputs=[instruction_input, readback_input], outputs=output)
    print(cj_model.summary())

    cj_model.compile(loss='mean_squared_error', optimizer='adam', metrics=['acc'])
    return cj_model
```

C.2 The Zhang & Wallace Architecture

```
    def build_model(self):
        print("\n—— Create Zhang & Wallace Network ———\n")
        instruction_input = Input(shape=(self.num_words, self.embedding_size,)
                            ,dtype='float32', name='instruction_input')
        instruction2 = Conv1D(300, 3, activation='relu', input_shape=(self.num_words,
                            self.embedding_size))(instruction_input)
        instruction3 = Conv1D(300, 4, activation='relu', input_shape=(self.num_words,
                            self.embedding_size))(instruction_input)
        instruction4 = Conv1D(300, 5, activation='relu', input_shape=(self.num_words,
                            self.embedding_size))(instruction_input)
        instruction2 = GlobalMaxPool1D()(instruction2)
        instruction3 = GlobalMaxPool1D()(instruction3)
        instruction4 = GlobalMaxPool1D()(instruction4)
        instruction_concat = concatenate([instruction2, instruction3, instruction4])

        readback_input = Input(shape=(self.num_words, self.embedding_size,)
                            ,dtype='float32', name='readback_input')
        readback2 = Conv1D(300, 3, activation='relu', input_shape=(self.num_words,
                            self.embedding_size))(readback_input)
        readback3 = Conv1D(300, 4, activation='relu', input_shape=(self.num_word,
                            self.embedding_size))(readback_input)
        readback4 = Conv1D(300, 5, activation='relu', input_shape=(self.num_words,
                            self.embedding_size))(readback_input)
        readback2 = GlobalMaxPool1D()(readback2)
        readback3 = GlobalMaxPool1D()(readback3)
        readback4 = GlobalMaxPool1D()(readback4)
        readback_concat = concatenate([readback2, readback3, readback4])

        final = concatenate([instruction_concat, readback_concat])
        final = Dense(units=128, activation="relu", kernel_constraint=max_norm(3))(final)
        final = Dropout(0.3)(final)
        output = Dense(self.num_classes, activation='softmax')(final)
        zw_model = Model(inputs=[instruction_input, readback_input], outputs=output)
        print(zw_model.summary())
```

```
    zw_model.compile(loss='mean_squared_error', optimizer='adam', metrics=['acc'])
    return zw_model
```

C.3 The Novel Architecture

```
def build_model(self):
    print("\n--- Create Novel Network ---\n")
    instruction_input = Input(shape=(self.num_words, self.embedding_size,),
                    dtype='float32', name='instruction_input')
    instruction = Conv1D(300, 2, activation='relu', input_shape=(self.num_words,
                    self.embedding_size))(instruction_input)
    instruction = Conv1D(300, 2, activation='relu', input_shape=(self.num_words,
                    self.embedding_size))(instruction)
    instruction = Conv1D(150, 1, activation='relu', input_shape=(self.num_words,
                    self.embedding_size))(instruction)
    instruction = GlobalMaxPool1D()(instruction)

    readback_input = Input(shape=(self.num_words, self.embedding_size,),
                    dtype='float32', name='readback_input')
    readback = Conv1D(300, 2, activation='relu', input_shape=(self.num_words,
                    self.embedding_size))(readback_input)
    readback = Conv1D(300, 2, activation='relu', input_shape=(self.num_words,
                    self.embedding_size))(readback)
    readback = Conv1D(150, 1, activation='relu', input_shape=(self.num_words,
                    self.embedding_size))(readback)
    readback = GlobalMaxPool1D()(readback)

    cos_sim = dot([instruction, readback], axes=1, normalize=True)
    final = concatenate([instruction, cos_sim, readback])
    final = Dense(units=128, activation="relu", kernel_constraint=max_norm(3))(final)
    final = Dropout(0.3)(final)
    output = Dense(self.num_classes, activation='softmax')(final)
    novel_model = Model(inputs=[instruction_input, readback_input], outputs=output)
    print(novel_model.summary())
    novel_model.compile(loss='mean_squared_error', optimizer='adam', metrics=['acc'])
    return novel_model
```

C.4 Model Evaluation

```
def evaluate_model(self, x1_test, x2_test, y_test):
    score = self.model.evaluate([x1_test, x2_test], y_test)
    acc = score[1] * 100
    loss = score[0]
    print("Accuracy on test data Novel Model: %0.1f" % acc + "%")
    print("Loss on test data Novel Model: %0.2f" % loss)

    label_pred_test = self.model.predict([x1_test, x2_test])
    max_y_pred_test = np.argmax(label_pred_test, axis=1)
    max_y_test = np.argmax(y_test, axis=1)

    print('Novel Model Scores')
    precision, recall, fscore, support =
    precision_recall_fscore_support(max_y_test, max_y_pred_test)
    print('Precision: {}'.format(precision))
    print('Recall: {}'.format(recall))
    print('F score: {}'.format(fscore))
    print('Novel Model Confusion Matrix')
    print(confusion_matrix(max_y_test, max_y_pred_test))

    m = multilabel_confusion_matrix(max_y_test, max_y_pred_test, labels=[0, 1, 2])[0]
    print(m)
    print('Erroneous Recall:')
    print(m[0, 0] / (m[0, 0] + m[0, 1]))
    return (acc, loss)
```

D Experimental Results

In this part of the appendix, the raw results from the experiments were consolidated and presented. Each model was run 15 times on different training and test splits. From the confusion matrices the precision, recall and f-score were tabulated for each class label. The bolded values are averages across the 15 runs. Macro and weighted averages were computed to inspect significance of class imbalance. Additionally, from the second confusion matrix the binarized recall score for the 'erroneous' class was derived and tabulated. This binarized recall is the statistic used to make a comparison against human error rates.

Table 9: Cheng & Jia Results

Precision			Recall			F-Score			Binarized Recall
Correct	Incomplete	Wrong	Correct	Incomplete	Wrong	Correct	Incomplete	Wrong	Erroneous Class
0.763	0.567	0.208	0.838	0.767	0.073	0.799	0.652	0.108	0.547
0.717	0.771	0.462	0.980	0.560	0.080	0.828	0.649	0.136	0.537
0.738	0.741	0.533	0.960	0.667	0.107	0.835	0.702	0.178	0.450
0.745	0.713	0.667	0.967	0.747	0.040	0.841	0.730	0.075	0.530
0.746	0.730	0.605	0.929	0.740	0.153	0.828	0.735	0.245	0.507
0.736	0.824	0.448	0.956	0.593	0.173	0.832	0.690	0.250	0.587
0.746	0.671	0.769	0.940	0.760	0.067	0.832	0.713	0.123	0.490
0.727	0.731	0.346	0.953	0.653	0.060	0.825	0.690	0.102	0.483
0.734	0.627	0.692	0.940	0.673	0.060	0.825	0.650	0.110	0.560
0.728	0.630	0.417	0.944	0.647	0.033	0.822	0.638	0.062	0.480
0.742	0.624	0.286	0.940	0.720	0.013	0.829	0.669	0.025	0.507
0.771	0.692	0.323	0.911	0.720	0.133	0.835	0.706	0.189	0.527
0.717	0.784	0.625	0.958	0.653	0.100	0.820	0.713	0.172	0.533
0.766	0.659	0.294	0.924	0.760	0.067	0.838	0.706	0.109	0.513
0.709	0.722	0.154	0.951	0.640	0.013	0.812	0.678	0.025	0.520
0.739	**0.699**	**0.455**	**0.939**	**0.687**	**0.078**	**0.827**	**0.688**	**0.127**	**0.518**
	Macro:	0.631		Macro:	0.568		Macro	0.547	
	Weighted:	0.674		Weighted:	0.717		Weighted:	0.659	

Table 10: Zhang & Wallace Results

Precision			Recall			F-Score			Binarized Recall
Correct	Incomplete	Wrong	Correct	Incomplete	Wrong	Correct	Incomplete	Wrong	Erroneous Class
0.808	0.863	0.556	0.909	0.587	0.527	0.856	0.698	0.541	0.750
0.844	0.754	0.539	0.864	0.613	0.600	0.854	0.676	0.568	0.693
0.788	0.786	0.675	0.944	0.687	0.360	0.859	0.733	0.470	0.670
0.803	0.815	0.688	0.940	0.707	0.427	0.866	0.757	0.527	0.680
0.777	0.809	0.712	0.958	0.733	0.280	0.858	0.769	0.402	0.720
0.813	0.810	0.700	0.947	0.680	0.467	0.875	0.739	0.560	0.670
0.810	0.793	0.677	0.940	0.713	0.420	0.870	0.751	0.519	0.717
0.798	0.770	0.701	0.942	0.780	0.313	0.864	0.775	0.433	0.663
0.819	0.861	0.595	0.942	0.580	0.520	0.876	0.693	0.555	0.693
0.807	0.759	0.592	0.909	0.733	0.387	0.855	0.746	0.468	0.673
0.848	0.747	0.588	0.882	0.727	0.533	0.865	0.736	0.559	0.747
0.770	0.865	0.608	0.953	0.553	0.393	0.852	0.675	0.478	0.700
0.815	0.858	0.680	0.947	0.727	0.453	0.876	0.787	0.544	0.663
0.824	0.829	0.619	0.907	0.713	0.520	0.863	0.767	0.565	0.723
0.782	0.798	0.671	0.951	0.660	0.353	0.859	0.723	0.463	0.717
0.807	**0.808**	**0.640**	**0.929**	**0.680**	**0.437**	**0.863**	**0.735**	**0.510**	**0.699**
	Macro:	0.752		Macro:	0.682		Macro	0.703	
	Weighted:	0.774		Weighted:	0.781		Weighted:	0.767	

Table 11:

Precision			Recall			F-Score			Binarized Recall
Correct	Incomplete	Wrong	Correct	Incomplete	Wrong	Correct	Incomplete	Wrong	Erroneous Class
0.862	0.853	0.841	0.947	0.813	0.633	0.903	0.833	0.722	0.773
0.878	0.813	0.852	0.927	0.867	0.653	0.902	0.839	0.740	0.813
0.872	0.793	0.773	0.904	0.867	0.613	0.888	0.828	0.684	0.800
0.866	0.781	0.690	0.889	0.760	0.653	0.877	0.770	0.671	0.793
0.857	0.785	0.897	0.942	0.827	0.580	0.897	0.805	0.704	0.763
0.872	0.691	0.789	0.862	0.880	0.600	0.867	0.774	0.682	0.810
0.859	0.855	0.855	0.944	0.827	0.627	0.899	0.841	0.723	0.767
0.850	0.850	0.760	0.933	0.720	0.653	0.890	0.780	0.703	0.753
0.869	0.800	0.707	0.902	0.800	0.627	0.885	0.800	0.664	0.797
0.861	0.744	0.784	0.909	0.813	0.580	0.884	0.777	0.667	0.780
0.863	0.787	0.822	0.920	0.887	0.553	0.890	0.834	0.661	0.780
0.862	0.836	0.802	0.956	0.780	0.593	0.906	0.807	0.682	0.770
0.853	0.921	0.746	0.944	0.700	0.687	0.897	0.795	0.715	0.757
0.873	0.796	0.732	0.913	0.807	0.620	0.893	0.801	0.671	0.800
0.854	0.750	0.658	0.858	0.760	0.640	0.856	0.755	0.649	0.780
0.863	**0.804**	**0.780**	**0.917**	**0.807**	**0.621**	**0.889**	**0.803**	**0.689**	**0.782**
	Macro:	0.816		Macro:	0.782		Macro	0.794	
	Weighted:	0.835		Weighted:	0.836		Weighted:	0.832	