BACHELOR THESIS

ARTIFICIAL INTELLIGENCE



|  |
| --- |
| Adaption and Analysis of Wav2Vec2 on Air Traffic Control Communication Data |

*First supervisor:*

dr. L.F.M. ten Bosch

Taal en communicatie

[louis.tenbosch@ru.nl](mailto:louis.tenbosch@ru.nl)

*Second supervisor:*

Dr. A. B. LAST NAME

Department

E-mail

*Author:*

Karan Chand

Student number: s1033357



28 August 2022

**Abstract**

The abstract of your thesis is a brief description of the research hypothesis, scientific context, motivation, and results.

The size of an abstract is usually one paragraph or one page of text.

* Recent work
* domain shift
* data evaluation
* xlsr model – why -> focus on multilingual, so accents good?
* wer and cer reduction
* study impact of size of data on finetune performance
* impact of arpa (maybe on other finetuned models as well)

Contents

[Introduction 4](#_Toc112073833)

[Preliminaries 6](#_Toc112073834)

[Related work 7](#_Toc112073835)

[Methodology 8](#_Toc112073836)

[Experiments 12](#_Toc112073837)

[Data 12](#_Toc112073838)

[Settings 12](#_Toc112073839)

[Results and Discussion 13](#_Toc112073840)

[Conclusions 16](#_Toc112073841)

[Bibliography 17](#_Toc112073842)

[Appendix 18](#_Toc112073843)

# Introduction

The introduction of your bachelor thesis introduces the research area, the research hypothesis, and the scientific contributions of your work.

The following things are usually in an introduction:

* Describe the problem and state the research question;
* Motivate why this problem must be solved;
* Demonstrate that a (new) solution is needed;
* Explain the intuition behind your solution;
* Motivate why / how your solution solves the problem (this is technical);
* Explain briefly how it compares with related work.

The introduction is closed with a paragraph in which the content in the next chapters is briefly mentioned to give an overview. A guideline is one sentence per chapter.

When travelling, the safety of the passengers depends on the drivers of the vehicle. For example, in trains and in airplanes communication between the drivers/pilots and the traffic control is crucial. Natural language processing can be used to make their job easier and less error prone by transcribing the natural language. Call signs for vehicles and directions are essential and errors should be minimized. Automatic Speech Recognition (ASR) is a perfect fit for this problem, as the transcriptions should be made in real time. There are many different models for ASR, but there is still a gap in the domain of Air Traffic Control (ATC).

Sequence-to-sequence models are commonly used for ASR applications, as they handle sequences of data well and can predict future data. Artificial Neural Networks (ANN), Recurrent Neural Network (RNN) and Hidden Markov Models (HMM) have among others been used for ASR(Trentin & Gori, 2001) (Zuluaga-Gomez et al., 2020). The state-of-the-art transformer architecture has been shown to give a lower Word Error Rate (WER) than RNN based algorithms. WER is the metric for the percentage of words that are incorrectly classified. Transformer models normally only work when a complete sentence/phrase is used as input. This is because the encoder/decoder transformer model predicts the current word using the previous and the following words. The transformer model has been adjusted(Moritz et al., 2020) to be able to use this in an online setting. Using the transformer in real-time can be done by time restricted self-attention on the encoder and triggered attention for the encoder-decoder mechanism. This way, the architecture can be used for ATC communication. To help ATC, online ASR is required to quickly process the communication between pilots and controllers. ATC can benefit from this architecture, because communication errors can be very costly. Reducing read-back errors in ATC have been researched before using an RNN and Long Short-Time Memory networks(JIA et al., 2018)(Lin et al., 2021). These models are able to get a relatively low WER, however, we should always aim for a lower WER, as it can potentially save lives.

Wav2vec2.0 performs well quote

Mistakes in Air Traffic Control ATC communication. Models for automatic speech recognition ASR exist, but constant improvement is needed. ASR models should minimize errors, which is still massively in progress. Minimizing errors in communication will reduce the risk of accidents.   
Models of ASR are being designed and updated constantly, which leads to great innovation in this field. An example of a community that specializes in Wav2Vec2 transformer models and data sharing is HuggingFace. Here, individuals can create, modify and publish their own models and datasets for others to use. Big companies such as Meta have shared their models as well, which are trained on massive amounts of data. These models are very recent and have been tested on ATC inadequately. (Has been done before though). In this paper, Wav2Vec2 transformer models are evaluated on their performance on transcribing ATC communication from the atcosim dataset. The models that are considered contain:

* A model that was pretrained on LibriSpeech data (Base)
* A model that was pretrained and fine-tuned on multiple noisy datasets as well as clean data (Robust)
* The finetuned XLS-R model in addition to a language model
* (Finetuned model on ATC data)

Technical ASR modifications on the models were not researched, as it was not the aim of paper.

The collection of ATC data is challenging and costly due to noise conditions, data privacy, rate of speech and language accent.

**Research questions:**

* How robust are pretrained and fine-tuned ASR models on ATC data?
* How does a language model affect the performance of a finetuned XLS-R model on ATC data?
* How does the amount of data used to finetune affect performance of a finetuned XLS-R model on ATC data?

# Preliminaries

This is an optional chapter that contains knowledge that your reader needs to know in order to understand your work. If there is not much you can also embed the preliminaries in one of the other chapters or even discard it completely.

* Transformer models
* HuggingFace
* what is fine-tuning
* what is pretrained

# Related work

Within this chapter you show that you have sufficient knowledge of the problem domain.

* Other models of ATC and their performance
* Wav2Vec2 Model and their performance
* Attention is all you need paper
* XLS-R on accents

# Methodology

**HuggingFace**  
Numerous python libraries exist for the development of ASR, such as scikit-learn and NLTK. These are standalone packages that are highly useful for tuning hyperparameters and developing NLP. For this research, complex development is not needed, rather testing of existing models is the focus. Testing of new and popular models can be more difficult to implement and test in these environments, as they are not easily accessible yet.   
The platform HuggingFace (HF) is more accessible. The HF community provides a wide range of publicly accessible models and datasets that can be modified with ease and reuploaded to the HF hub for public use. Over 25 transformer tasks are available for use, e.g. ASR, image classification and text generation. Models and datasets are constantly updated, which implies that state-of-the-art tools are attainable. HF has a course for beginners and a decent documentation of the main libraries used: Datasets and Transformers. The learning curve is high, but the models and data delivered outweigh the costs compared to other libraries.

**Development**Different platforms are used for the development of transcribing, evaluating, training and testing of ASR models. HF recommends using Google Colab for ease of programming and the absence of owning a high-end GPU for computation. For low computational power demanding tasks, such as small batch transcription, Google Colab is used, but the free to use platform has limitations that are diminishing our resources. GPUs can be either T4, P100 or a K80. These GPUs are inconsistently allocated by Google Colab and are unpredictable. Fine-tuning models and transcribing full datasets regularly exceed the resources (GPU ram, disk space and system ram) made available by Google Colab, leading to resource deprivation.   
For this reason, local development in scripts is done in addition to cloud development. A GeForce GTX 1080 GPU is used for transcribing full datasets and training of models for more robust development.

**Dataset**  
There is a distinct shortage of publicly available data for ATC that can be used for ASR, as most datasets do not include transcriptions quote. Websites such as ‘*liveatc.net*’ provide, as the domain name implies, live and very recent ATC communication audio. Based on airport codes and frequencies, different radio communication is selected and listened to. This data does not include transcriptions and is therefore not suitable for training our models. Audio data is abundant and publicly available; however, audio data has to be manually transcribed, which is costly and time-consuming, leading to lower accessibility of transcribed ATC data.   
In this research, the ATCOSIM dataset is used. This corpus provides ten hours of speech data, pronounced by ten non-native speakers in the English language. A subset of the dataset can be found in the appendix. It includes additional information on the recording sessions and speakers, such as: Gender, speaker id, recording length, etc. This information was not used/ researched as it is out of the scope of this paper.  
The ATCOSIM dataset is a simulation of real-time ATC. The data is considered clean, as speech recordings were made in a controlled environment (Hofbauer et al., 2018). Transcriptions have an average duration of 3.8 seconds. The same ATC lingo is used in the 10-hours of speech data. The data is noisy, as a close-talk headset microphone is used. Speaking style, language use, background noise and stress levels were kept as realistic as possible, but the representability of ATCOSIM to real ATC data has not been studied yet. This corpus has been chosen anyway, as variables that exists in real ATC speech, such as microphone quality, are not the focus of research here. However, this should be taken into consideration when evaluating the generalisability of the studied ASR models.

The perplexity of the transcriptions of ATCOSIM is compared to the Google/fleurs corpus from HF. The Google/fleurs dataset contains sentences that are much more common in regular speech than the ATCOSIM dataset. The mean perplexity of the test cases from the Google/fleurs dataset is 35.98, while the ATCOSIM bears a staggering 175.08 mean perplexity. This shows that the ATCOSIM significantly diverges from normal speech and constitutes to low performance of regular speech models on this data.

The ATCOSIM dataset can be downloaded from the internet and can be obtained on DVD. It is not yet available on HF. The corpus is formatted in ISO, which needs to be reformatted for HF transformers use. As mentioned above, additional information on speakers and recordings are removed from the dataset and will not be used.  
Additionally, unusable rows in the dataset are removed. Unusable meaning:

* Rows containing transcriptions that exceed the maximal duration allowed for training and model transcription, as GPU usage is a limitation in this research.
* Rows containing information that is chosen not to be included for research as it is either out of the scope or to increase efficiency of ASR. Such information includes: Human noise, word fragments, empty utterances, off-talk, nonsensical words, foreign language and unknown words. These are represented by the tags respectively: [HNOISE], [FRAGMENT], [EMPTY], (<OT> … </OT>), [NONSENSE] and (<FL> … </FL>). Where three dots represent the speech that corresponds to the tags.
* The characters in the transcription that are regarded as not useful for training and transcribing. Keeping the transcriptions as simple as possible generally increases efficiency of ASR models quote. Characters that were removed, separated by ‘*\*’: [*\=\~\@\,\?\.\!\-\;\:\"\“\%\‘\”\�\'*].

The ATCOSIM corpus originally has 10078 rows, however after cleaning up and pruning, 9397 rows remain. The train-validation-test split is made using an 8:1:1 ratio, which correlates to 7526-940-931 rows respectively. However, due to resource limitations, only a maximum of 1000 training rows and a constant 50 validation rows are used for training. The number of test cases stay consistently 931.

**Models**  
The models used are imported from the HF community. They are not trained or fine-tuned specifically on ATC data, as that domain has not been explored/published by the community yet. It is important that the models are recent, as ASR models are consistently updated and improved upon. Four models were considered for evaluating Wav2Vec2 models:

Base: facebook/wav2vec2-base-960h.   
Currently the most popular ASR model on HF that is updated approximately every 3 months. It has been pretrained and fine-tuned on 960 hours of Librispeech data. This model is called ‘base’ as it is mostly used for further training on a downstream task (finetuning). This model works well when clean, regular speech data is fed.

Robust: facebook/wav2vec2-large-robust-ft-swbd-300h.   
This model is a pretrained and fine-tuned version of the facebook/wav2vec2-large-robust model. The latest update to this model was 5 months ago. The data it has been fine-tuned on 300 hours of the Switchboard corpus. This is a telephone speech corpus that contains noisy data. This model has been chosen to see exactly how robust a robust model is when transcribing ATC data.

huBERT: facebook/hubert-large-ls960-ft.   
This model has been fine-tuned on 960 hours of Librispeech data and had its currently last update three months ago. The huBERT model follows the Wav2Vec2 architecture and extends upon it. huBERT’s training process is fundamentally different from Wav2Vec2, which the specifics of are out of scope. It has been shown that the model either improves or matches the performance of Wav2Vec2 (Hsu et al., 2021), so it is interesting to see if it performs better than the base and robust model on transcribing ATC data.

XLS-R: jonatasgrosman/wav2vec2-large-xlsr-53-english.  
The XLS-R model has been used for learning multilingual speech representation, dialect identification and accent identification (Pascal & Dominique, 2021). These challenges are apparent in ATC transcription as well, even though they are not the main objectives. The XLS-R model has been chosen, because the architecture uses contrastive learning and might perform better compared to the other models as a result. This specific model is updated regularly, with the last update pushed to HF being less than a month ago since beginning research.

**Evaluation**  
Evaluation of the model transcriptions was done by using the Word Error Rate (WER) and Character Error Rate (CER). These are the standard for ASR models as the accuracy from speech to text is the most important. Metrics such as Glue or Bleu are not useful here, as they give information on the meaning of the output text compared to the reference text. These metrics are not needed in our ASR models and are thus omitted.  
The perplexity metric is used as well to determine how rare sentences are to appear in a gpt-2 text generation model. This model is trained on the text on 45 million Reddit web pages and contains 1.5 billion parameters. The gpt-2 model has been chosen, because it is the most popular text generation model found on HF with over 12 million downloads.

|  |  |
| --- | --- |
| **Training steps** | **Hours** |
| 10 | 0.011 |
| 50 | 0.053 |
| 150 | 0.158 |
| 500 | 0.528 |
| 1000 | 1.056 |

**Fine-tuning**   
Fine-tuning (training) is done using the standard template given by user ‘patrickvonplaten’ on the HF blog. Hyperparameters are not tuned to the dataset and models, as resources are limited. Training is done locally, as Google Colab has limitations that are mentioned above.   
The number of training data steps in which the models were fine-tuned in are chosen from the three defined distributions: low, medium and high data ranges. Ranges being from 1-100, 101- 500 and 501-inf respectively. Due to time and resource restrictions, only five models are fine-tuned.

* Two models in the low data range: 10 and 50 training samples.
* Two models in the medium data range: 150 and 500 training samples.
* One model in the high data range: 1000 training samples.

*Table 1: Ranges converted to hours*

**Addition** **Language** **model**  
The language model was made using an ARPA file containing only the training and validation transcriptions. Excluding the test cases from this file improves integrity and reduces overfitting. Overfitting is already a problem, as the dataset is quite small and only contains 743 unigrams. Because the word count is low, 1-5-grams are stored. Higher/lower N-grams are available, but are not studied here. Including a language model can boost the performance of ASR and is tested in this paper.

**XLS-R model**The reasoning behind choosing to finetune only the XLS-R model consists of three reasons. Firstly, the model is evaluated to be the most interesting to research on ATC data, as it has been shown that the model handles accents and multilingual data very well in comparison to the other models (Babu et al., 2021). Secondly, this model is exceedingly recent, as updates are posted regularly. It could have been interesting to fine-tune all the models; however, our resources would not grant this, producing the third reason. Given this information, the XLS-R model is the best option.

# Experiments

## Data

## Settings

All training experiments use the same set of hyperparameters. The feature encoder is frozen during the fine-tuning phase, as it is trained sufficiently during pretraining. We fine-tune each model for 7.5k steps, with a 100-step warm-up phase. We fine-tune each model on a GeForce GTX 1080 with an effective batch size of 8 (batch size of 4, gradient accumulation of 2). Learning rate is increased linearly until 3e−4 during warm-up, then it linearly decays.

# Results and Discussion

* ~~Performance of pretrained models~~
* ~~Performance of finetuned models~~
* ~~Performance of finetuned model with language model~~
* ~~Performance of xls-r model over versions~~
* Errors best model makes

First, the pretrained models that are directly pulled from HF are evaluated. In figure 1 it shown that the WER and CER are relatively high. These models were not given an explicit language model. The base model performs the worst, which is expected, as it has not been pretrained on noisy speech data. It has the lowest WER and CER. The base model is the oldest model that is researched, which should be noted.   
The robust model performs better compared to the base model, as it is pretrained and fine-tuned on noisy telephone speech data. This results in a significant reduction in WER and a decent reduction in CER.  
The huBERT model possesses the smallest WER and CER. Improved or identical performance compared to standard Wav2vec2 models is expected from the huBERT model (Hsu et al., 2021). A modified architecture might be the cause of this, as the model has been fine-tuned on the same data as the base and robust model.  
The XLS-R model was expected to have a lower WER than the base and robust model, as it ought to have an advantage in transcribing data that has multiple accents.   
The CER is relatively low for all models, which indicates that the models are able to recognize letters being spoken, however, they do not have a language model specific to the data. Airport names and callsigns are uncommon in regular speech, relating to a high WER.

Figure 1: Evaluation of pretrained models in WER and CER.

To continue, the XLS-R model is fine-tuned on increasing amounts of data (figure 2). We see a massive drop in WER and CER as the amount of data increases. The WER Reduction (WERR) for the XLS-R-10 model is already at ~39%. The highest WERR is seen found in the XLS-R-1000 model at ~93%. CER Reduction CERR for the XLS-R-1000 model is significant as well (~92%).

Figure 2: Evaluation of the fine-tuned XLS-R models in WER and CER.

Next, a language model is added to the Wav2Vec2 processor. We can see a further reduction of the WER and CER. The model with the lowest WER and CER is, as expected, the XLS-R-1000\* model. However, the WERR of the lower data provided models are significant and impressive as well.

Figure 3: Evaluation of the fine-tuned XLS-R models in addition to a language model in WER and CER.

The overall improvement of the XLS-R model is shown in figure 4. It shows that fine-tuning on ATC data can significantly improve the overall, especially in addition to a language model.

Figure 4: Improvement of the XLS-R model in WER and CER.

**Comparison to Zuluaga-Gomez et al. 2022**  
Similar work on fine-tuning older models has been done on ATC data (Zuluaga-Gomez et al., 2022). Zuluaga-Gomez and colleagues’ lowest utilization of training data consists of 5 minutes of speech and achieved WERs of 40% and 43.9% for ISAVIA and NATS respectively. Air navigation service providers collected and annotated these datasets, which originated from actual ATC communication. As a different dataset is used for evaluation, a comparison between the models does not seem logical. Especially since the data that is used here is not actual ATC data. However, it should be mentioned that the WER and CER acquired from the XLS-R-1000\* model are extremely low, considering only 1 hour of training data was used to fine-tune the model.  
The lowest utilization of training data in our research is 38 seconds and achieved a WER of 43% without an addition language model. Addition of a domain specific language model further decreases WER to 0.29, resulting in a better ASR model using less training data. Again, these model performances should not be compared to the work that has been done by Zuluaga-Gomez and colleagues, rather it should give insight on the model capability.

**Unusually high performance - Overfitting**  
The definite WERR and CERR of the XLS-R-1000\* model compared to the original pulled xls-r model from HF amount to ~95.5% and ~96.1% respectively. Such high rates are unusual and could be explained by the origins of the ATCOSIM dataset. As it is simulated, data is similar. Ten speakers are used to fill the corpus and recordings have the same audio quality.  
Also, all speakers were integrated in the train, test and validation set. This can lead to overfitting; therefore, it would have been better to isolate speaker(s) from the train and test set during preprocessing.

**Evaluation XLS-R-1000\***  
Evaluate errors - arpa errors, why those errors and should we have a different metric than WER and CER? Command error rate may be better.  
Exact -> expect – unigrams but no context, so harder to predict. Not really that bad, as callsigns are more important

# Conclusions

In this chapter you present all conclusions that can be drawn from the preceding chapters.

It should not introduce any new material or theories; these should have been written down earlier in the thesis. Because of this, your conclusion can be brief and to the point.

When further research?

* Fine tune on different models, was not the focus of this paper
* Try finetuned models on more data for comparison and generalisation factor
* Finetune on more data or on different training settings, as hyperparameters have not been tuned for our ATC data.
* Add LM to lower finetuned models to see the performance difference
* Train on more samples using better resources
* Different arpa files and n-gram sizes
* Tune hyperparameters for training
* Finetune with better steps

# Bibliography

Babu, A., Wang, C., Tjandra, A., Lakhotia, K., Xu, Q., Goyal, N., Singh, K., von Platen, P., Saraf, Y., Pino, J., Baevski, A., Conneau, A., & Auli, M. (2021). *XLS-R: Self-supervised Cross-lingual Speech Representation Learning at Scale*. http://arxiv.org/abs/2111.09296

Hofbauer, K., Petrik, S., & Hering, H. (n.d.). *The ATCOSIM Corpus of Non-Prompted Clean Air Traffic Control Speech*.

Hsu, W. N., Bolte, B., Tsai, Y. H. H., Lakhotia, K., Salakhutdinov, R., & Mohamed, A. (2021). HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units. *IEEE/ACM Transactions on Audio Speech and Language Processing*, *29*, 3451–3460. https://doi.org/10.1109/TASLP.2021.3122291

JIA, G., CHENG, F., YANG, J., & 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. https://doi.org/10.1016/j.cja.2018.10.001

Lin, Y., Guo, D., Zhang, J., Chen, Z., & Yang, B. (2021). A Unified Framework for Multilingual Speech Recognition in Air Traffic Control Systems. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(8), 3608–3620. https://doi.org/10.1109/TNNLS.2020.3015830

Moritz, N., Hori, T., & le Roux, J. (2020). *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

Pascal, F., & Dominique, R. (2021). *Bachelorarbeit (Informatik) Speech Classification using wav2vec 2.0*.

Trentin, E., & Gori, M. (2001). A survey of hybrid ANN/HMM models for automatic speech recognition. In *Neurocomputing* (Vol. 37).

Zuluaga-Gomez, J., Prasad, A., Nigmatulina, I., Sarfjoo, S., Motlicek, P., Kleinert, M., Helmke, H., Ohneiser, O., & Zhan, Q. (2022). *How Does Pre-trained Wav2Vec2.0 Perform on Domain Shifted ASR? An Extensive Benchmark on Air Traffic Control Communications*. http://arxiv.org/abs/2203.16822

Zuluaga-Gomez, J., Veselý, K., Blatt, A., Motlicek, P., Klakow, D., Tart, A., Szöke, I., Prasad, A., Sarfjoo, S., Kolčárek, P., Kocour, M., Černocký, H., Cevenini, C., Choukri, K., Rigault, M., & Landis, F. (2020). *Automatic Call Sign Detection: Matching Air Surveillance Data with Air Traffic Spoken Communications*. 14. https://doi.org/10.3390/proceedings2020059014

# Appendix

Appendices are optional chapters in which additional material is covered. This material is required to fully support your research, that would otherwise clutter the presentation of your research.

An subset of the ATCOSIM dataset:  
  
More information on the dataset can be found at: <https://www.spsc.tugraz.at/databases-and-tools/atcosim-air-traffic-control-simulation-speech-corpus.html>