Student Internship Programme (SIP)

Final Project Report

at

DSO National Laboratories

Reporting Period:

05 2021 to 07 2021

by

Chew Cheng Yap

Department of Computer Science

School of Computing

National University of Singapore

2019/2020

Project Title: Punctuation Restoration in speech-to-text transcripts
Project ID: SY20274251
Project Supervisor: Prof Hsu, Wynne

Summary

Restore missing punctuation in speech-to-text transcripts. We want to obtain a model which makes use of as little training data as possible and achieve relatively good results. This is because we might not have an abundance of labelled data when using it during application.

Acknowledgements

This project had been supervised by Chieu Hai leong and Yap Yong Keong from DSO National laboratories. This project is also largely built on a study (Alam, 2020) by Tanvirul Alam, Akib Khan and Firoj Alam, Punctuation Restoration using Transformer Models for High-and Low-Resource Languages.

Table of Contents

a. Introduction

Main task for this internship: Restore missing punctuation in speech-to-text transcripts. We want to obtain a model which makes use of as little training data as possible and achieve relatively good results. This is because we might not have an abundance of labelled data when using it during application.

Punctuation restoration models are usually trained on clean texts but used on noisy Automatic Speech Recognition (ASR) texts and thus performance may be reduced due to errors which are not present in the training data. A current state-of-the-art study (Alam, 2020) makes use of transformer-based models and proposed an augmentation strategy which manages to obtain comparable state-of-the-art results. The augmentation strategy aims to address the issue above and manages to yield a 3.8% relative improvement in the F1 score on ASR transcriptions for English and obtains state-of-the-art results. In my experiments, I will use this augmentation strategy as the default implementation when training on my datasets

b. Datasets

IWSLT dataset

The IWSLT dataset consists of transcriptions from TED talks. The training and development set consist of 2.1M and 296K words, respectively. Two test sets are provided with manual and ASR transcriptions, each containing 12626 and 12822 words, respectively. This dataset is used in the above study in which I will also use the same splits for my experiments.

Switchboard dataset

1155 5-minute conversations that are manually labelled from the Switchboard corpus of telephone conversations. The splits I use changes throughout the course of my internship.

Originally, I have a split which simply splits the whole continuous file into 3 files resulting in a 1.1M, 294K, 163K set split. Next, the training, dev and test sets are sampled by conversations, and then merged into single files resulting in 220K train, 500K dev, 912K test sets in number of words.

The LJ Speech Dataset

This is a public domain speech dataset consisting of 13,100 short audio clips of a single speaker reading passages from 7 non-fiction books. A transcription is provided for each clip. Clips vary in length from 1 to 10 seconds and have a total length of approximately 24 hours. The train,

dev and test sets were split by sampling blocks of 1000 words, resulting in 45K train, 16K dev and 160K test sets in number of words.

c. Experimental Settings

I build on the existing model architecture used in the study (Alam, 2020) which allows for different pre-trained models used for transfer learning. I used mainly 3 pre-trained models: bert-base-uncased, bert-base-multilingual and xlm-roberta-base, available in the HuggingFace's Transformers library with their model-specific tokenizers for tokenization.

Each sequence length will be 256, which starts with a special *start of sequence* token and ends with an *end of sequence* token. I use the default batch size of 8 and the sequences will be shuffled before each epoch during the training phase. I also use the optimum values from the study to augment my datasets during training.

The datasets used vary from experiment to experiment, however within each experiment, only the heuristic is applied and tested. I make use of the F1 scores across the 3 models to have an estimate on the impact of each heuristic.

d. Experiments

Full sentence inputs

Originally, the inputs are passed in as tokens sequentially in order without checking if it truncates part of a sentence. This proposed method feeds in each sequence of length 256 by ensuring that each sequence does not contain parts of more than 1 sentence during processing.

There was a general slight increase in F1 scores across all models, however, after consulting with my supervisors, this method introduces a bias as our model does not know the start and ends of sentence during application which renders this experiment unsuccessful. Employing this method requires careful consideration of our use case and our underlying assumptions about out inputs.

Training data concurrently vs Saving a model for transfer learning

We make use of a big dataset and a small dataset from two different distributions. First, we train the models on both datasets together and concurrently. On the other hand, we train the model on the big dataset then save the model, before restoring and training it again with the smaller dataset.

The latter significantly degrades the performance when tested with the distribution from the smaller dataset. Since we want to do well on this smaller target distribution, we continue to train data from different distributions concurrently rather than separating them.

Sliding window strategy

We propose a sliding window strategy to be used during training and testing to increase the number of training data and utilize our data more effectively. We then apply a weight function to the predicted values to ensure that all tokens are equally considered despite their duplication. Having a stride size of 0.5, we can modify the weight function to garner different results. The weight function multiplies each predicted value by a scalar value, and we want to reduce the values at the ends of each sequence as each sequence block may truncate parts of a continuous sentence, it will be less useful as there is lesser room for interpretation. Figure 1 shows an illustration of the training process with this implementation.

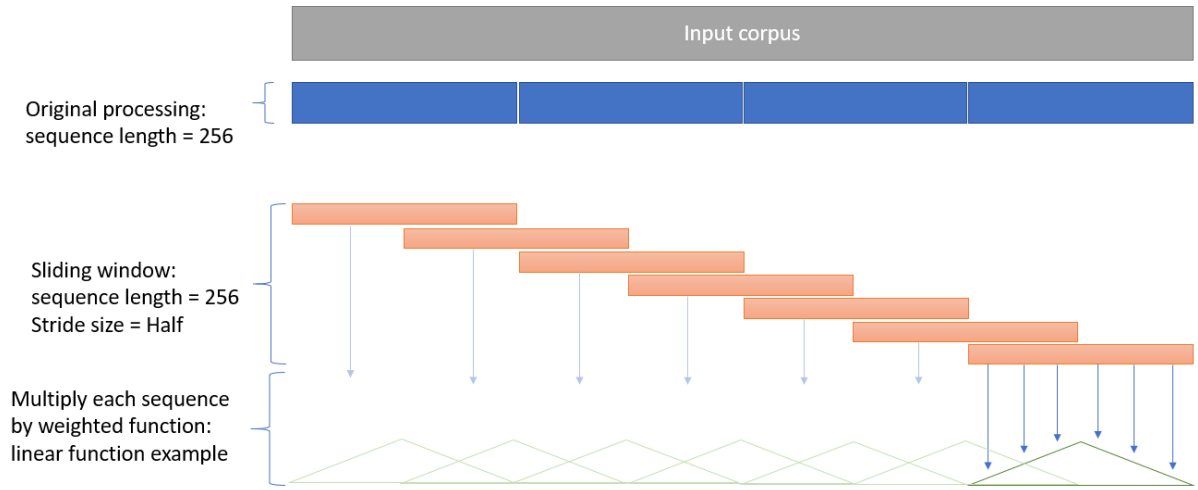

Figure 1

For validation and testing, we need to sum every overlapping sequence since now there would be 2 prediction values for each token.

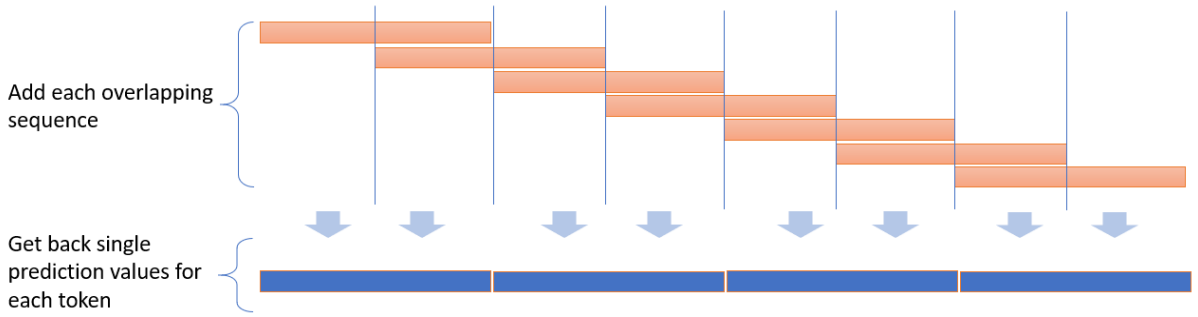

Figure 2

The dataset used for this experiment is 2.1M words from the IWSLT dataset together with 220K words from the Switchboard dataset. Its performance is then tested on the 912K words Switchboard dataset. Below shows the distribution of the punctuations.

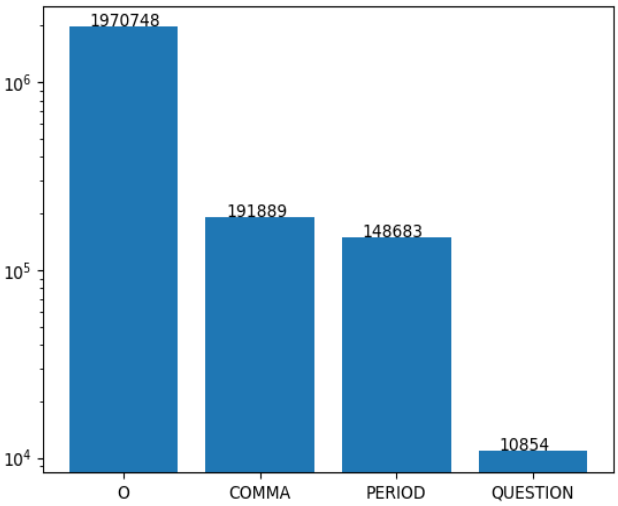

Figure 3, Train: 2.1M IWSLT + 220K Switchboard

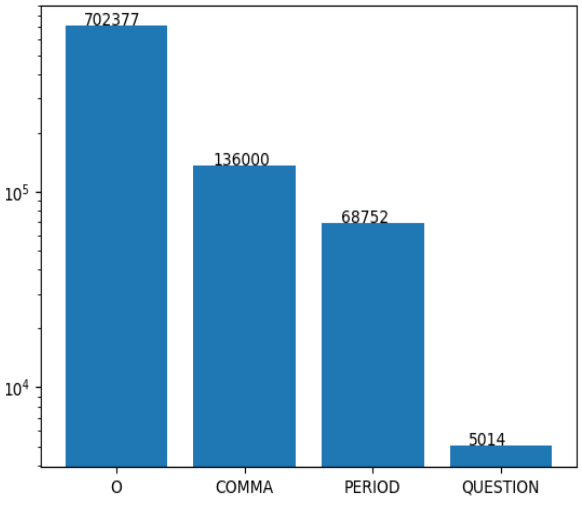

Figure 4, Test: 912K Switchboard

3 different weight functions are used in this experiment and F1 scores are consolidated below:

(1) a linear function where the end weights for each sequence is 0.5

(2) a linear function where the end weights for each sequence is 0

(3) a sin squared function where the end weights for each sequence is 0

Model	Test	No Sliding window	Sliding window (1) (0.5 to 1 linear function)	Sliding window (2) (0 to 1 linear function)	Sliding window (3) (Sin Squared graph)
Bert Base uncased	912K Switchboard	0.749	0.762	0.762	0.761
Bert base multilingual uncased		0.746	0.760	0.756	0.758
Xlm Roberta base		0.750	0.770	0.770	0.769

Table 1: F1 scores on models trained on 2.1M IWSLT + 220K Switchboard

From table 1, there is an average of 1.6% improvement when using the sliding window where we reduce the weights nearing the ends by half. This gives the best result compared to the other forms of the weight function.

Two other distributions of data are used to cross check the effects of the sliding window implementation. For both, I used the 1.1M words Switchboard dataset as the base training data. I then train it with the smaller datasets from the other 2 distributions (IWSLT and LJ Speech) respectively and tested it on the larger set of their respective distributions. The following shows the distributions of the train and test set followed by the table of F1 scores.

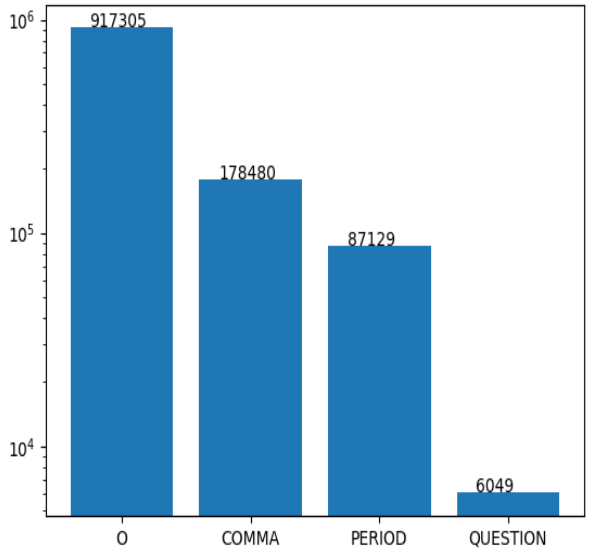

Figure 5, Train: 1.1M Switchboard + 12.6K IWSLT

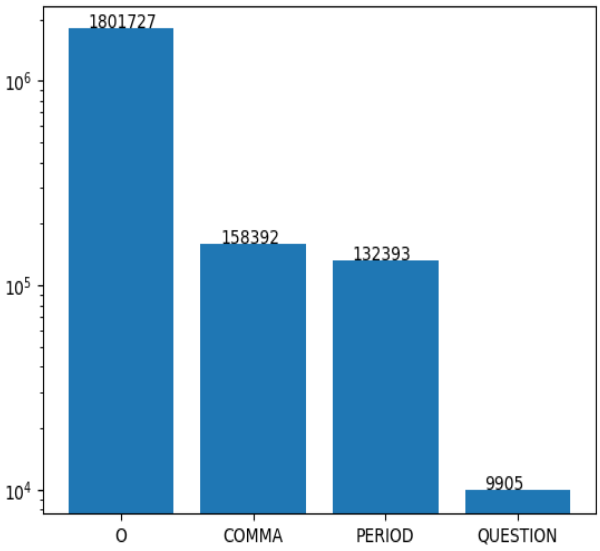

Figure 6, Test: 2.1M IWSLT

Model	Test	Default implementation	Sliding window (1) (0.5 to 1 linear function)
Bert Base uncased	2.1M IWSLT	0.600	0.620
Bert base multilingual uncased		0.604	0.633
Xlm Roberta base		0.653	0.676

Table 2: F1 scores on models trained on 1.1M Switchboard + 12.6K IWSLT

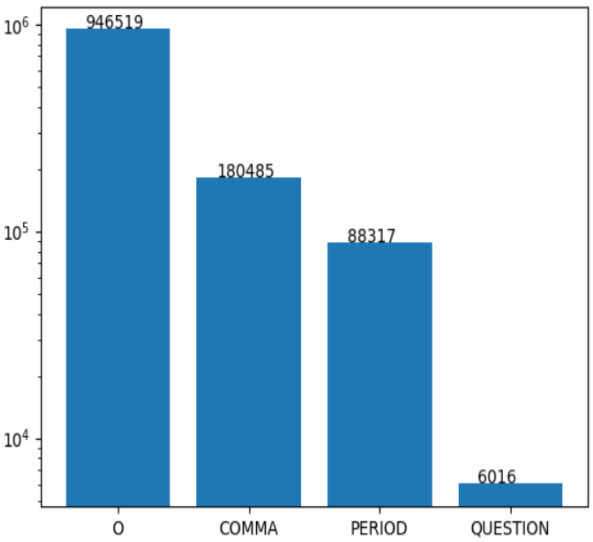

Figure 7, Train: 1.1M Switchboard + 45K LJ Speech

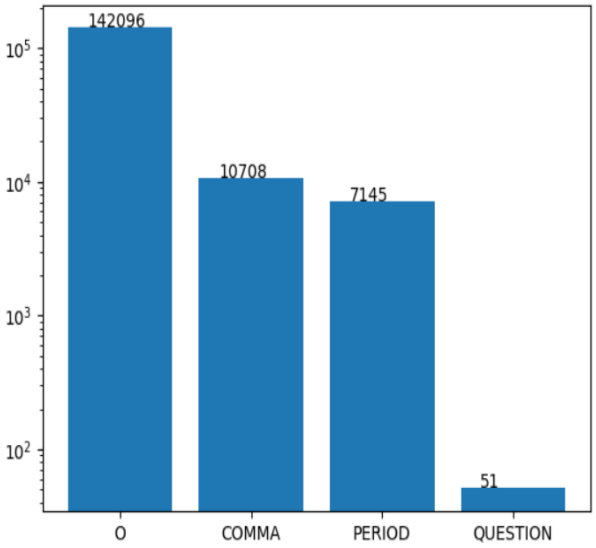

Figure 8, Test: 160K LJ Speech

Model	Test	Default implementation	Sliding window (1) (0.5 to 1 linear function)
Bert Base uncased	160K LJ Speech	0.641	0.704
Bert base multilingual uncased		0.657	0.720
Xlm Roberta base		0.695	0.749

Table 3: F1 scores on models trained on 1.1M Switchboard + 45K LJ Speech

We can see that the other distributions yield a more significant improvement. Using only 12.6K words of data for training for the IWSLT dataset, we can improve the score by 2 to 3% when testing on a 2.1M set. Using only 45K words of data for training for the LJ Speech dataset, the F1 scores improved by an average of 6% across all models.

Sliding window strategy on individual input files

The above strategy was applied on contiguous blocks of conversations of the Switchboard dataset, I attempted to apply this strategy on individual conversations rather than have sequences contain parts from different conversations, however there was no significant change in results.

Weighing the loss function

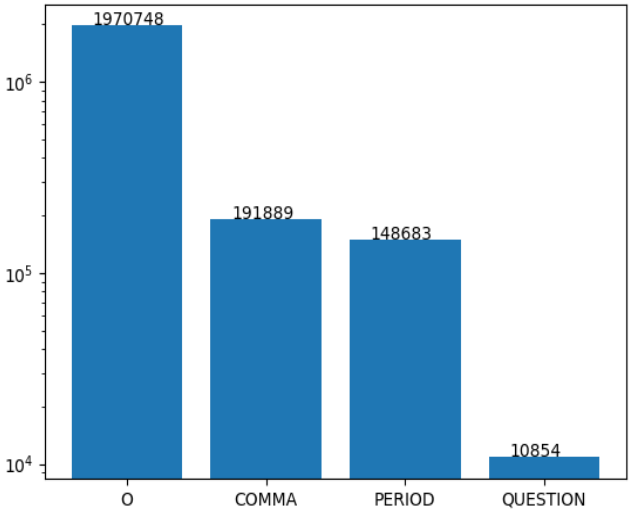

Figure 9, Example train distribution from Switchboard dataset

From figure 9, we can see that the dataset is skewed and that there are much lesser question marks, for example, compared to the rest of the punctuations. So, weighing the loss function would mean punishing the model more for making a wrong prediction for such punctuations with small relative quantity.

We trained and tested using the best sliding window implementation, (1) from above on its same distribution and compare the results to it summarized in the table below. 3 different weight functions are constructed with its result (weights of 'O', 'COMMA', 'PERIOD', 'QUESTION') as inputs for the loss function:

(1) Subtracting the percentage of the punctuation from 1: [0.15, 0.91, 0.93, 0.99]

(2) Taking fractions inverse of all the punctuations scaled to 1: [0.0048, 0.049, 0.064, 0.88]

(3) Taking the logarithm of the fractions inverse of all the punctuations: [0.16, 2.49, 2.74, 5.36]

Model	Test	Sliding window (1) (0.5 to 1 linear function)	(1) [0.15, 0.91, 0.93, 0.99]	(2) [0.0048, 0.049, 0.064, 0.88]	(3) [0.16, 2.49, 2.74, 5.36]
Bert Base uncased	912K Switchboard	0.762	0.740	0.714	0.706
Bert base multilingual uncased		0.760	0.729	0.707	0.701
Xlm Roberta base		0.770	0.748	0.718	0.721

Table 4: F1 scores on models trained on 2.1M IWSLT + 220K Switchboard

There was a decrease in test performance when using the additional weight function to change the loss values for all the weight functions. It was found that there was a low precision score on the punctuations with lesser quantity while their recall scores remain high. This suggests that the model does not want to fail to predict such punctuations and ended up overcompensating such punctuations.

Other Improvements

Making inputs lower case

The speech-to-text transcripts to be tested on are given as lowercase characters. The pre-trained models offer uncased versions which converts text to lowercase before WordPiece tokenization, however we can manually convert them during data processing without relying on the model selection.

Removing start, end and pad tokens before passing it into the loss function

When calculating the accuracy, precision, and F1 scores, a mask is used to ensure that only the predictions for every word is accounted for. In the loss function, there is no such mask and removing the start, end and pad tokens would offer a more accurate representation of the loss values from words only.

Increasing the number of epochs

At the early stages, the number epoch was set to 10 by default and the best model is updated based on the highest validation accuracy on the validation set. However, after plotting the validation accuracy at every epoch, I realised that the value did not peak and increasing the number of epochs could result in getting a better model. The number of epochs has been increased to 15 and training will halt if validation accuracy does not increase after 3 consecutive epochs.

e. Results and Discussions

To get the best results on a small dataset, it is still necessary to train the model with a small sample from the same distribution, this is to account for a large difference in distributions from the larger dataset.

For the full sentence inputs, we can use this heuristic depending on how the inputs are being processed. For example, if your use case processes sequences based on pauses in speech, we

may assume the inputs are in sentences. For this project, the datasets used does not have these assumptions hence it will introduce a bias to the model.

The sliding window strategy does improve the F1 scores in general. I additionally tested this implementation on 2 models(bert base uncased, bert base multilingual uncased) with the exact distribution of data from the study (Alam, 2020) and ran the same tests as given by the IWSLT dataset. There was an average improvement of 0.7% on ASR transcriptions for English and 2% on manual transcriptions.

When we weigh the loss functions in our models, we should check the individual recall and precision scores as well, rather than just the overall F1 score. This could provide us with more information and in our case above, we found that the model overcompensated the predictions of those punctuations in low quantities.

f. Secondary Task

The switchboard dataset has by far been processed and formatted to be split up by conversations and speakers. It has been processed in another way by the sensors(sr) department, segmenting the data into time segments. The train and dev segments from their new split results in shorter segments from parts of a conversation compared to whole speaker conversations. The new data distribution does not come with punctuations and we want to restore them using the original dataset, we now face an alignment issue where we want to restore punctuations to this new dataset which has different noise from the existing datasets. Below is a side-by-side comparison of the two datasets we are trying to align.

Sr data:	Original:
sw02006-B_042685-043291 exactly and that's um when you start m- when you start paying your wages you know jerry takes a different outlook towards you guys	Exactly. And that's, um, when, when you start, when you start paying your way, uh, you know, Jerry takes a different outlook towards you guys.
Legend: Insertions Deletions Substitutions	

Bag of words approach

We first locate all the punctuations of the original dataset and save its nearest 13 words. We then systematically go through every 13-word block in the sr data which gives the best match for that punctuation. This method relies heavily on how well we can find the best matching position to restore the punctuation.

By inspection, this method seems to give relatively good results with a few errors shown below.

Sr data	Original
1. but, you know, people are buying them. you know whose buying them? the power company. 2. free, you know, free for them. 3. kind of interesting so see. 4. uh-huh. 5. yeah it's. top speed is something like eighty miles an hour. so it's a good, good vehicle. 6. uh-huh 7. oh, i wasn't there too long ago. 8. well, you know, y-, going over the hills, you know, coming into the valley? you can see that horrible, horrible brown haze.	But, you know, people are buying them. You know whose buying them? The power company. Free, you know, free for them. Kind of interesting so see. Uhhuh. Yeah <child>. It's top speed is something like eighty miles an hour. So it's a good, good vehicle. <child_talking>. Uhhuh <talking>. Oh, I wasn't there too long ago. Uh. Well, you know, yo, going over the hills, you know, coming into the valley? <child_talking>. You can see that horrible, horrible brown haze.

In addition, when tested with the model trained on 2.1M words from the IWSLT dataset together with 220K words from the Switchboard dataset (Page 3) using the sliding window strategy, there was only a slight decrease of 0.01 to 0.015 in F1 scores. This could suggest that this approach restores punctuations relatively well.

Model Inference approach

This approach uses my existing model to restore punctuations to the sr dataset. I first train the model with the corresponding original dataset. However, this method does not give as good results compared to the bag of words method. An example of the same excerpt is shown below with more errors.

Sr data	Original
1. but, you know, people are buying them, you know, whose buying them, the power company free, you know, free for them. kind of interesting. so see. uhhuh. yeah, it's top speed is something like eighty miles an hour. so it's a good, good vehicle. uhhuh. oh, i wasn't there too long ago. well, you know, y, going over the hills, you know, coming into the valley, you can see that horrible, horrible brown haze	But, you know, people are buying them. You know whose buying them? The power company. Free, you know, free for them. Kind of interesting so see. Uhhuh. Yeah <child>. It's top speed is something like eighty miles an hour. So it's a good, good vehicle. <child_talking>. Uhhuh <talking>. Oh, I wasn't there too long ago. Uh. Well, you know, yo, going over the hills, you know, coming into the valley? <child_talking>. You can see that horrible, horrible brown haze.

Challenges faced during your internship

As I did not have much prior experience with machine learning, there were many new concepts and foreign terms such as the different types of models and networks used for different purposes. It was daunting at first, but thankfully I had some background in Python and computing which helps me learn such concepts quickly. With proper guidance from my supervisors, I am able to leverage and build upon an existing model and get on my task quickly.

Lessons learnt from the tasks assigned to you

Machine learning tasks deal with lots of data and there are many aspects to keep track of. It is crucial to meticulously keep track of each goal and target as data can easily be mixed up and errors can be unaccounted for. Documentation is something that I can improve on especially when starting out on a new project in the future. It also helps when consolidating data for presentation purposes.

I should start with clear documentation of code as well as results in the future, it makes it easier to understand my work both for myself as well as others. One thing I can do in the future is keep track of all the hyperparameters that I have changed for every experiment so that I do not lose track and have a clear focus on my objectives.

Areas to improve

Ability to speak with people in a professional setting and working independently with little guidance. Presenting ideas clearly and succinctly during meetings.

Having the end goal in mind is important and it is good to stay organised when dealing with data. Documenting and ability to understand my work in the future is equally important as getting the job done. It also serves to keep me in check so that I do not stray off while experimenting on a research project.

I should prepare the necessary tools required and ensure that I can explain my ideas clearly, especially if it requires many changes and lines of code. I can also improve by adopting some of the best practices when dealing with machine learning projects in the future as I have learnt from my supervisors.

References

Alam, T. a. (2020). *Punctuation Restoration using Transformer Models for High-and Low-Resource Languages.* Association for Computational Linguistics. doi:10.18653/v1/2020.wnut-1.18