# PM2.5 forecast with Seq2Seq model

Xiang Yao Ng
Electronic Engineering, Tsinghua University
Beijing, China
huang-xy19@mails.tsinghua.edu.cn

## Abstract

*Air pollution is a global problem that negatively affects people's lives. It is in the general consensus of most people that air pollution should be curbed to the minimum to maintain the people's health. Hence, the need to monitor and predict air quality indices are crucial so that actions can be taken to keep pollution from rising unexpectedly. In this paper, I will explore the Seq2Seq model to perform PM2.5 forecast.*

***Keywords:*** Air pollution, PM2.5, LSTM, Seq2Seq

## 1. Introduction

Air pollution is an environmental problem that plagues the health and wellbeing of the masses in a society. It occurs when harmful substances are released into Earth's atmosphere. Sources of air pollution include sulfur dioxide (SO2), nitrogen dioxide (NO2), ozone (O3), and carbon monoxide (CO). To prevent air pollution from rising to dangerous levels, we need to constantly monitor and regulate the air pollutants. One index to monitor the air quality is PM2.5. PM2.5 refers to atmospheric particulate matter (PM) that have a diameter of less than 2.5 micrometers.

The task of PM2.5 forecast is to monitor and measure past PM2.5 data from air quality monitoring stations and give a prediction of future PM2.5 values based on local emission sources and remote sources (pollutants carried by wind from afar). Air pollution forecasting is often done by coupling weather forecasting systems with atmospheric dispersion modelling. Machine learning algorithms are also employed to learn good models so as to enhance air quality prediction. Some machine learning algorithms that are popular are the Autoregressive integrated moving average (ARIMA), Vector Auto-Regressive (VAR), Gradient Boosting Regression Tree (GBRT), LSTM, Seq2seq and etc.

In this paper, I will demonstrate the Seq2Seq model commonly used in the Natural Language Processing (NLP) field to make PM2.5 forecast. I used past 24 hours of feature data to predict the next six hours of PM2.5.

## 2. Data Preparation

There are a lot of missing data in the given air quality data. While some of the missing data can be substituted with mean imputation, it remains a daunting task as sometimes data of all monitoring stations for particular times are lost. Hence, the data needs to be cleaned before it can utilised. For this purpose, I have set some rules to filter and fix the data for the forecast:

1. Remove CSV with whole rows of lost data.
2. Some CSVs does not have proper data structure of 24 hours and 35 monitoring stations when reading the CSVs. However, it is very important to retrieve these data in a proper structure so that accurate description of the air quality at its specific time and venue can be obtained. Hence, all CSVs that does not fit this requirement is removed.
3. For my model, I concatenate two subsequent days of data. Therefore, if no two subsequent days of CSV is found, the lone data will be excluded.

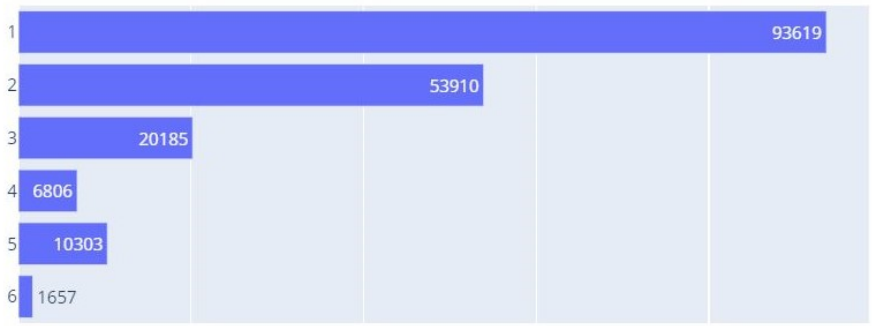
| Years | Air Quality | Air Pollutants |
|---|---|---|
| 2014 | 318 | 237 |
| 2015 | 350 | 350 |
| 2016 | 334 | 334 |
| 2017 | 245 | 245 |
| 2018 | 306 | 306 |
| 2019 | 306 | 306 |
| 2020 | 120 | 120 |

Table 1. Number of CSVs in dataset

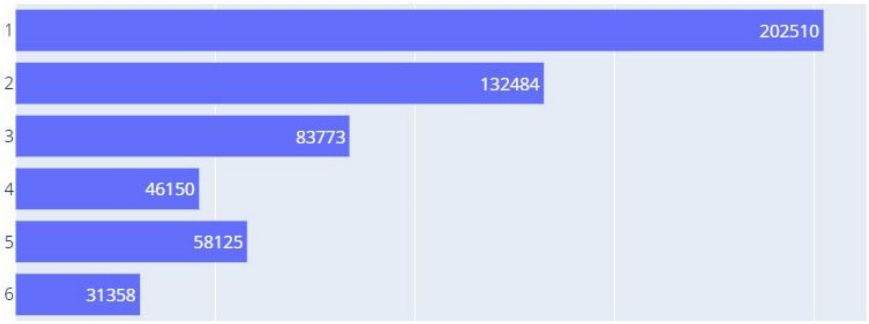
After filtering the data, the number of CSVs for air quality (such as PM2.5) and air pollutant (such as SO2, CO etc) is shown in Table 1. A distribution has also been plotted to visualize the class counts of PM2.5 of each year in Figure 1.

## 3. Feature Selection

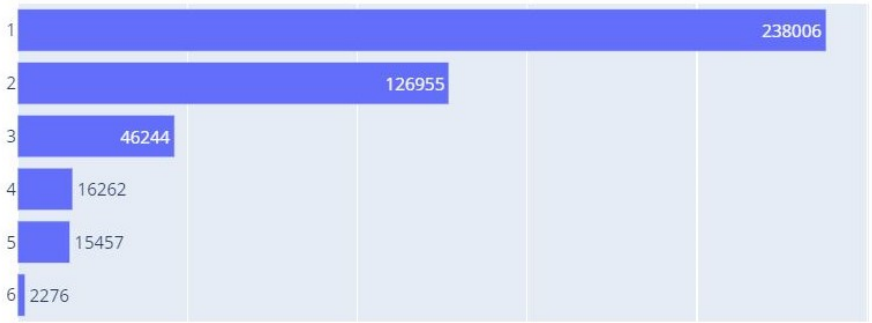
Possible feature data that are available from the given dataset is PM2.5, PM10, AQI air quality indices, as well as SO2, NO2, CO, and O3 air pollutants. In my model, I grouped up air pollutants as feature data after normalizing each air pollutants data to $[0-1]$. I picked only PM2.5 as another feature data. This is because PM10 is an indicator of different size compared to PM2.5, hence it is plausible to think that in some cases where PM2.5 can be very high, PM10 may not rise to the same level as PM2.5, which may affect the prediction results. On the other hand, the AQI has a distribution that changes according to the range of the PM2.5. For



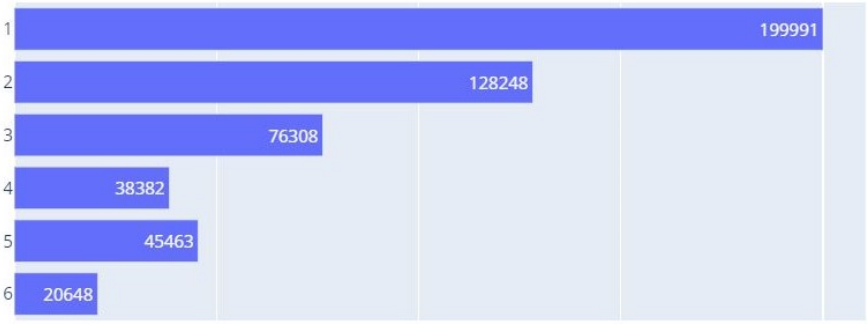

(a) PM2.5 2014



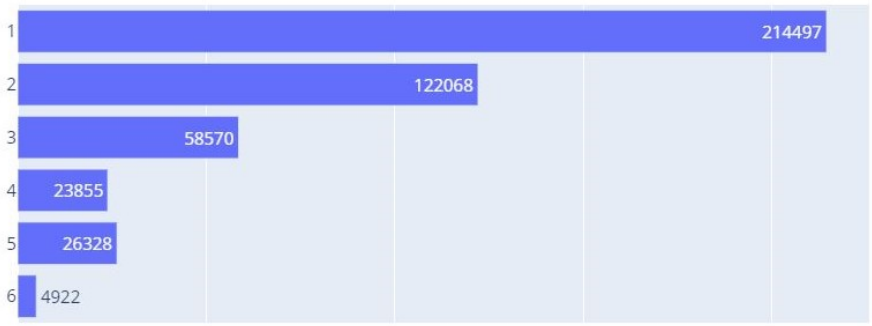

(b) PM2.5 2015



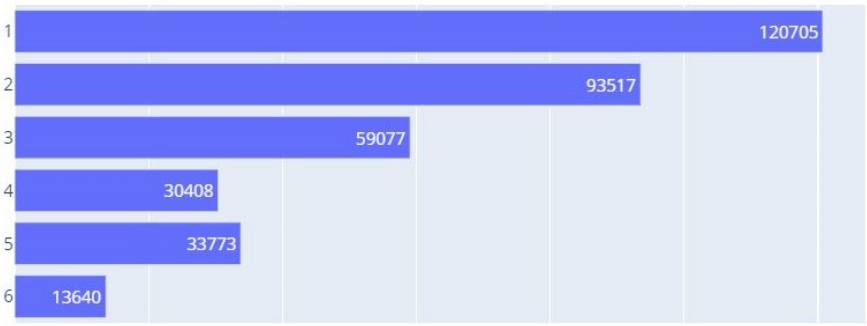

(c) PM2.5 2016



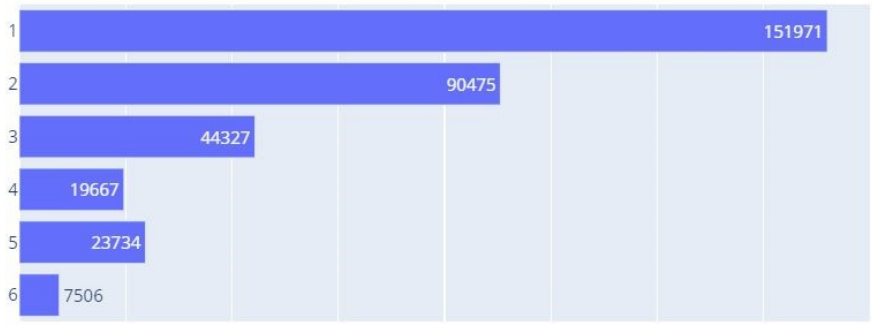

(d) PM2.5 2017




(e) PM2.5 2018




(f) PM2.5 2019




(g) PM2.5 2020

Figure 1. Class counts of PM2.5 from 2014 to 2020

| Weights | Class Accuracy (in decimals) | | | | | | Mean Accuracy |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | |
| [3.0, 3.0, 1.5, 1.0, 1.0, 0.5] | 0.803 | 0.197 | 0.0 | 0.0 | 0.0 | 0.0 | 0.469 |
| [1.0, 1.0, 1.0, 1.0, 1.0, 1.0] | 0.964 | 0.023 | 0.0 | 0.0 | 0.0 | 0.0 | 0.500 |
| [0.005, 0.1, 0.3, 0.5, 0.7, 1.0] | 0.015 | 0.855 | 0.123 | 0.0 | 0.0 | 0.0 | 0.277 |
| [0.05, 0.05, 1, 3, 3, 5] | 0.876 | 0.0 | 0.140 | 0.0 | 0.042 | 0.0 | 0.464 |
| [0.005, 0.005, 1, 3, 3, 10] | 0.440 | 0.0 | 0.653 | 0.0 | 0.0 | 0.0 | 0.294 |

Table 2. Variation of weights to class accuracy

instance, for China's AQI conversion to PM2.5, an PM2.5 increase of 0.7 is accompanied by an AQI increase of 1 between range 0-35 of PM2.5. However, it changes to 0.8 PM2.5 per AQI in the 35-75 range of PM2.5. Hence, this irregularity is not very ideal for the forecast.

After the mentioned considerations, the final feature selection is the combination of air pollutants data and the PM2.5. The shape of data structure of air pollutants data and PM2.5 are

$[(num\_csv \times 4), 24, 35]$ and $[(num\_csv), 48, 35]$ for each year. The 4 multiplier for air pollutants was due to the combination of the 4 air pollutants. For air pollutants, only the data from past 24 hours is retrieved while data for 48 hours is retrieved for PM2.5 so that the data for past 24 hours can be used to forecast the next 6 hours of PM2.5, as shown in Figure 2.

### 3.1. Model

The Seq2Seq model that I have used are referred from the PyTorch website [1]. It is a simple encoder-decoder architecture that takes in feature data as input for encoder and predict output by the decoder.

**Model Architecture** An example of the model architecture is shown below in Fig. 2. First, both air pollutants data and air quality data of the previous 24 hours are trained in two separate encoders. Then, both context vectors are combined and passed to the decoder for training. By using random teacher forcing, the next output is predicted using the previous output or the target itself .

**Encoder** The encoder is a simple one that takes the input into an LSTM and output hidden inputs. Given a sequence of 24 of input $x_{t-24;t}$, the hidden state is updated via a LSTM by the following equation:

$$h_t = f_e(h_{t-24}, x_{t-24;t})$$

where $f_e$ is the LSTM with a sequence length of 24.

**Combine context vectors** The combine part simply concatenates the context vectors together and then pass through fully connected layers to train them. Given hidden states $ha_t$, $hb_t$ as the hidden states from two encoders computed for air quality and air pollutants data respectively, the hidden states are combined by the following method:

$$hd_t = W_c[ha_t \; ; \; hb_t] + b_c$$

where $hd_t$ is the context vector that will be pass to the decoder, while $W_c$ and $b_c$ are parameters that needed to be learned from a fully connected layer.

**Decoder** As mentioned before, it was either the previous output and target that was used as input. The input will first go through one-hot encoding, which then passed to the LSTM for processing. Given decoder input $y_t$, the output and hidden states will be updated as follows:

$$y_t^{'}, hd_t^{'} = f_d(y_t, hd_t)$$

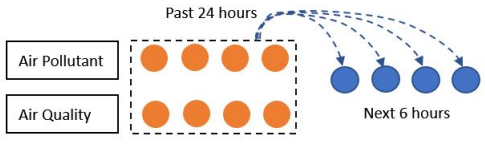



Figure 2. PM2.5 forecast

where $y_t^{'}$ is the output, and $f_d$ is a LSTM with sequence length of 6.

It will then be passed through a log-softmax as follows:

$$\gamma_t^i = log(\frac{exp(y_t^{'i})}{\sum_{j=1}^{T} exp(y_t^{'j})})$$

**Dataset imbalance problem** In order to tackle the dataset imbalance problem, where most class labels are from class 1 and 2, I tweak the weights to the loss function so that heavier penalties are imposed to classes with higher number of samples. This also makes it the hyperparameter.

**Attention Mechanism** I tried adding attention mechanism to my Seq2Seq model. However, slight increase in hidden units size causes big CPU ram intake, which causes the sessions to crash, while low hidden size negatively impact the effectiveness of training, causes training to fail. Due to time restriction, I do not have more time to change my original model. Therefore, I scrapped the model with the attention mechanism and keep the original one.

## 4. Experiment

### 4.1. Dataset split

For training data, the data structures are $[(num\_csv \times 4 \times 6), 24, 35]$ and $[(num\_csv \times 6), 48, 35]$ to account for data from 2015 to 2020. For test data, the shape of data structure of air pollutants data and PM2.5 are $[(num\_csv \times 4), 24, 35]$ and $[(num\_csv), 48, 35]$, which is taken from year 2014.

### 4.2. Hyperparameters

Some hyperparameters are fixed as I feel that changes on the hyperparameters does not affect the results. These hyperparameters are listed below:

1. $hidden\_size = 64$
2. $epoch = 50$
3. $learning\_rate = 0.001$

For my experiments, I vary the weights to the loss function to see the impact of class weights to the accuracy. The results are shown in the Table 2.

### 4.3. Best Model

Since the majority of samples are in class 1, the evaluation of the results should based on a wider spectrum of accuracy to test its effectiveness in predicting other classes of smaller samples instead of just the mean accuracy. After all experiments on change of hyperparameters, the best model is the one with weights $[0.05, 0.05, 1, 3, 3, 5]$. The training loss is plotted and the accuracy distribution is shown in Figure 4 and Figure 5 respectively.

## 5. Analysis

### 5.1. Dissection of Models

For this analysis, the encoder is removed to investigate the effectiveness of the decoder. The zero vector is now passed as initial

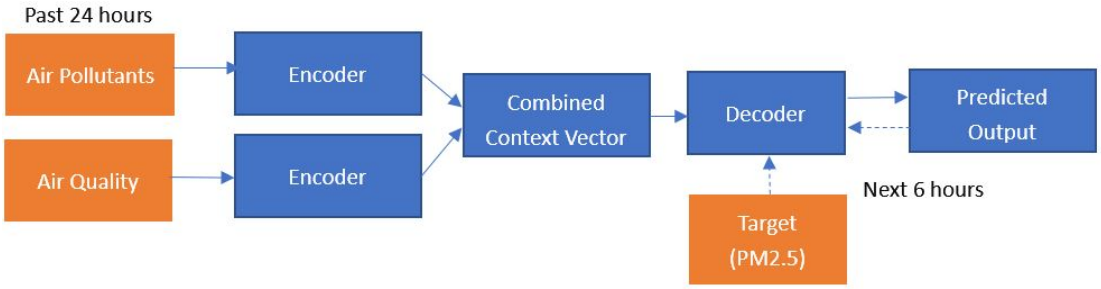



Figure 3. Model Architecture

| | Class Accuracy (in decimals) | | | | | | Mean Accuracy |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | |
| With Encoder | 0.876 | 0.0 | 0.140 | 0.0 | 0.042 | 0.0 | 0.464 |
| Without Encoder | 0.781 | 0.0 | 0.314 | 0.0 | 0.0 | 0.0 | 0.402 |
| With all data | 0.876 | 0.0 | 0.140 | 0.0 | 0.042 | 0.0 | 0.464 |
| Only 2016 and 2018 data | 0.911 | 0.0 | 0.116 | 0.060 | 0.0 | 0.0 | 0.479 |

Table 3. Analysis of encoders and datasets

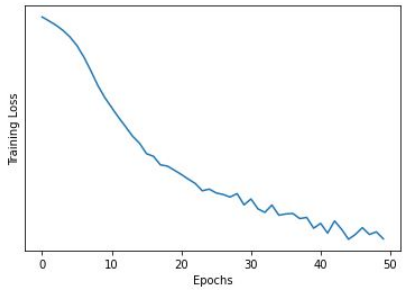



Figure 4. Training Loss of Best Model

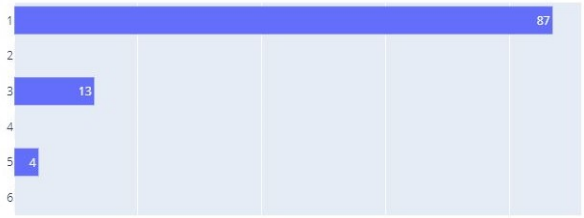



Figure 5. Accuracy distribution of Best Model

hidden units instead of the context vector generated by the encoder. The best model is tested for this experiment. The results are shown in Figure 3.

From the results, we can tell that the overall accuracy decreases when the encoders is excluded from the model. This is because information from encoders are not included to the computation, only the present ground truth data is used to predict future output.

### 5.2. Dissection of Datasets

Looking at the distributions of the classes alone, I decided to include the dataset of 2016 and 2018 data only because of the similarity of distributions between the former group and 2014. Results are shown in Table 3.

The reason why the results slightly improved is because training on data with similar distribution as the test data allows the models to fit the distribution of test data more smoothly.

## 6. Code Reference

Some of the codes are modified from the reference of Seq2Seq model in PyTorch [1] in train.py and test.py. The code will be submitted to the WebLearning Tsinghua portal, as well as the Github website `https://github.com/Ut0pi4/pm2_5_prediction`. There are two ways to describe how the code from other source was used. The term "taken" refers to codes used in my own code without any modification, while the term "modify" refers to codes used with a certain modification, often large modifications to suit the problem needs.

## 7. Discussion of the sudden change points of air quality index

Due to lack of time, I did not consider the detection of sudden changes of PM2.5. However, I would like to explain how I would do it if more time is given. First, I would have to modify the data retrieval to include sudden change points. If the increment of PM2.5 is more than a certain threshold, the sudden change point of a certain time at a certain monitor station is given a 1, otherwise 0. Once the data is retrieved, it can be used to train a separate model to identify the sudden change points, where the loss will be compounded with the Seq2Seq model to allow training.

## 8. Conclusion

We have explored the Seq2Seq model on performing PM2.5 forecast with air pollutant data and air quality data. Best model shows a level of diversiveness to the accuracy. I did not add mechanisms to predict sudden changes of PM2.5 simply because I took

too much time on designing the model architecture, which did not work out well for me. In the future, better time-series forecasting models can be explored to tackle this problem.

## References

[1] NLP from scratch: Translation with a sequence to sequence network and attention. https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html. Accessed: 2020-06-05. 4, 5