Fine dust prediction using Convloutional Long Short-Term Memory Model and geological matrix

Abstraction

This paper proposes a new prediction method for the increasing amount of ultrafine dust in Korea. Unlike predictions using existing simple statistics, we use matrix that take into account the geological location of China and Korea. In addition, LSTM(Long Short-Term Memory) was adopted to use existing time series fine dust data. The results of this method are compared with the existing data, and the problems to be solved in the future are suggested.

1 Introduction

The fine dust in Korea is increasing rapidly as the year goes by. Furthermore, as more and more small fine dust (PM 2.5) increases, it is hard to see with the naked eye.In this situation, prediction of fine dust is becoming very important.However, current methods of prediction are less accurate. There have been many studies to solve this problem.

Previous researches have mainly focused on one area, and some of them are based on the assumption of the reason for the high concentration of fine dust, and the reason for this is the relationship between the fine dust and the reason. However, as in many weather forecasts, there are many anomalies that can not be explained by reason alone.

So, we have predicted by building up a geological matrix consisting of fine dust except for the data and other additional factors for the last three years, deviating from predicting using decades old data which were mainly used.

2 Background and related work

Convolution In mathematics (and, in particular, functional analysis) convolution is a mathematical operation on two functions (f and g); it produces a third function, that is typically viewed as a modified version of one of the original functions, giving the integral of the pointwise multiplication of the two functions as a function of the amount that one of the original functions is translated. Convolution is similar to cross-correlation. It has applications that include probability, statistics, computer vision, natural language processing, image and signal processing, engineering, and differential equations.

RNN(Recurrent Neural Network) A recurrent neural network (RNN) is a class of artificial neural network where connections between units form a directed cycle. This creates an internal state of the network which allows it to exhibit dynamic temporal behavior. Unlike feedforward neural networks, RNNs can use their internal memory to process arbitrary sequences of inputs. This makes them applicable to tasks such as unsegmented connected handwriting recognition or speech recognition.

LSTM Long short-term memory (LSTM) is a recurrent neural network (RNN) architecture (an artificial neural network) proposed in 1997 by Sepp Hochreiter and Jürgen Schmidhuber and further improved in 2000 by Felix Gers et al. Like most RNNs, a LSTM network is universal in the sense that given enough network units it can compute anything a conventional computer can compute, provided it has the proper weight matrix, which may be viewed as its program. Unlike traditional RNNs, an LSTM network is well-suited to learn from experience to classify, process and predict time series when there are time lags of unknown size and bound between important events. Relative insensitivity to gap length gives an advantage to LSTM over alternative RNNs and hidden Markov models and other sequence learning methods in numerous applications. Among other successes, LSTM achieved the best known results in natural language text compression, unsegmented connected handwriting recognition, and in 2009 won the ICDAR handwriting competition. LSTM networks have also been used for automatic speech recognition, and were a major component of a network that in 2013 achieved a record 17.7% phoneme error rate on the classic TIMIT natural speech dataset. As of 2016, major technology companies including Google, Apple, Microsoft, and Baidu are using LSTM networks as fundamental components in new products. For example, Google uses LSTM for speech recognition on the smartphone, for the smart assistant Allo, and for Google Translate. Apple uses LSTM for the ”Quicktype” function on the iPhone and for Siri. Amazon uses LSTM for Amazon Alexa.

3 Experiment enviroment

The experimental environment is shown in Figure 1 and is an Intel i7-based 8-core machine. I used the GTX1080 as the GPU.

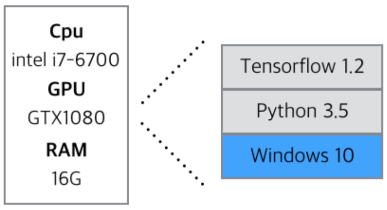

그림 1: Environment

For the measurement environment, software of the same version as Table 1 was used. And the input data is collected by using the data that is open to the Internet.

표 1: Software version and input data

OS	Tool	Python	Tensorflow	input data
Windows 10	Tensorflow	3.5.2	1.2.0	fine dust data

4 Fine dust prediction

This section explains how this differs from the existing rule-based approach and details its model. We modeled the existing methods and theories appropriately. The table 2 shows the name of the popular forecasting model, whether it is public or not, and information about the model name.

표 2: Existing model

Name	Public	Model name
Cams	No	Copernicus Atrmosphere Monitoring model
SILAM	Yes	System for Integrated model
SPRINTARS	Yes	Spectral Radiation-Transport model
PANDA	No	-

4.1 Approach

Most of the publicly available models address the causal relationship of how the actual fine dust is generated. But we decided to pay more attention to the results more simply. A two-dimensional matrix was generated based on the actual PM2.5 data. In this process, we tried to solve the relation between Chinese fine dust and Korean fine dust by borrowing not only Korean data but also Chinese data. We create a data matrix based on the day and plot it as follows Figure 2. Convolution is applied by applying a 3x3 kernel based on this 90x90 matrix. We try to predict by applying such inputs to the LSTM model. Our hidden layer size was 10, and the total number of steps was 760 days. Stride did not skip data at 1, and the optimizer used adamoptimizer to learn to minimize loss.

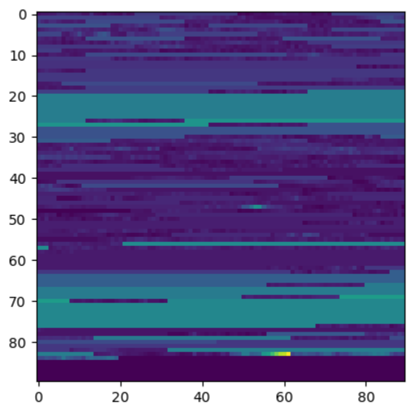

그림 2: Data matrix image

To illustrate the model, we can briefly explain this Figure 3 . Convolve the basic data in the upper left corner to create new data. It then learns through the recurrent network and creates new predictions with the learned weights.

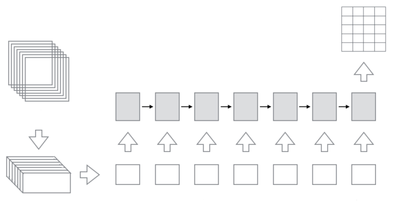

그림 3: Model summary

5 Conclusion and Future Work

We predicted a total of 7 matrix based on the model we trained and compared with the rest of the commonly used methods against existing measured data. You can see from the Table 3. Amazingly disappointing results came out. The predictability of all existing models is over 80%, while the model we trained has not even exceeded 65%. So I thought about what was lacking and there were various problems.

First, the data was too small. Our original plan was to split the 24 hour day into hours and model it. However, there was some

표 3: Result

Name	1day	3day	7day
Cams	89%	90%	90%
SILAM	91%	88%	85%
SPRINTARS	85%	87%	91%
PANDA	88%	85%	83%
Out Method	62%	52%	43%

damage to the acquired data, and there was also a problem with the coverage of the data, so We arbitrarily subdivided the sections to create the input data every day. As a result, the total data did not exceed 1000, and based on the year, the data was based on a little more than two years of data.

Second, We thought of too few factors when constructing the model. We thought that China was the biggest problem, and we thought about this model with the assumption that all the results would be right if we had an association with China. However, as a result, we noticed that there are many other factors as well as China, so we need to model it.

참고 문헌

[1] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. WardeFarley, and Y. Bengio. Theano: New features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[2] Y. Bengio, I. Goodfellow, and A. Courville. Deep Learning. Book in preparation for MIT Press, 2015.

[3] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In Scipy, volume 4, page 3. Austin, TX, 2010.

[4] R. Bridson. Fluid Simulation for Computer Graphics. Ak Peters Series. Taylor & Francis, 2008.