Natural Language Processing approach applied in Crime Predicting

Guangyi Chen

School of Computer Science , University of South Carolina

Abstract:

What The crime data in South Carolina from 2000-2020

Using entity extraction and LMST

The result achieved is crime number predicting

• Introduction:

The Natural Language Processing (NLP) system in this project will help process large textual

data relevant to criminal investigations by extracting specific information from cases and

reports. The system can potentially be applied on many other forms of unstructured data like

messages, letters, testimonies, court trials, and conversations. After heavily training the

system and reaching a decent level of correctness, it can be officially introduced as a main

step of the investigation process. It could save investigators a significant amount of time, as a

machine is significantly faster than a human and can provide more accurate and consistent

results with less chances of errors through Information Extraction (IE) , since it is not affected

by human factors such as fatigue or distraction. Text summarization, which is the secondary

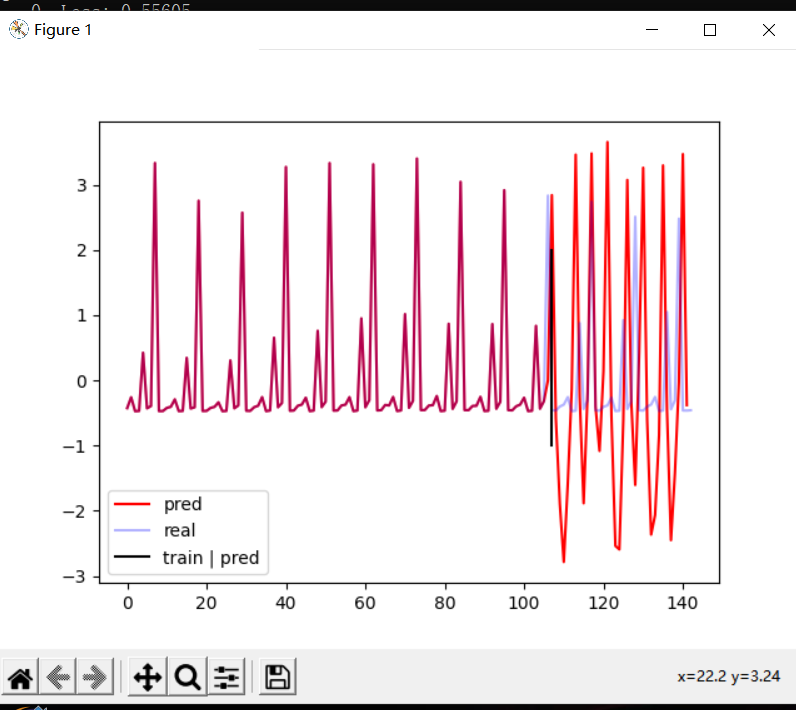
objective of this system, would synthesize all the information gathered into coherent text and

save time when building case files

• Problem // Clearly state input and output

Input: Crime type, City

Output: crime number rate tendency



• Related Work

Automatic Crime Prediction using Events

Extracted from Twitter Posts

Xiaofeng Wang, Matthew S. Gerber, and Donald E. Brown

Department of Systems and Information Engineering, University of Virginia

Crime forecasting: a machine learning and computer vision approach to crime prediction and prevention

Neil Shah, Nandish Bhagat & Manan Shah

Visual Computing for Industry, Biomedicine, and Art volume 4, Article number: 9 (2021)

Performance Evaluation of a Natural Language

Processing approach applied in White Collar crime investigation

Maarten Banerveld2, Nhien-An Le-Khac1 and M-Tahar Kechadi1

1School of Computer Science & Informatics, University College Dublin

Belfield, Dublin 4, Ireland

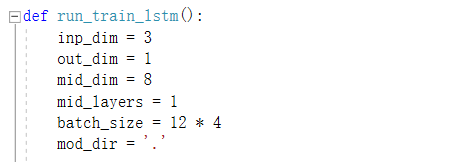
1 How does the system work

* 1. data set:

I put the data directly in the code. This set of data is nibrs\_criminal\_act , a total of 144 sets of data. The data of the first 9 years (75%) is used as the training set, and the data of the next 3 years (25%) is used as the test set. I normalized the data (subtract mean, divide by standard deviation).

1.2 Model

We hope to input the passenger flow data of the first 9 years and let LSTM predict the passenger flow of the next 3 years, then we can first train the LSTM to predict the passenger flow of the next month based on the data of the previous 9 years. After the training is completed, we let LSTM predict the passenger flow of the next month based on the data of the first 9 years, take the predicted passenger flow just output as input, and iteratively obtain the passenger flow of the next 3 years.



Then input the input.size() of this LSTM == (seq\_len, batch\_size, inp\_dim)

Then input the input.size() of this LSTM == (seq\_len, batThe length of seq\_len time series, here the first 9 years are 9\*12 == 108 sets , then seq\_len == 108 ch\_size, inp\_dim) because we have 12type of crime

Then input the input.size() of this LSTM == (seqinp\_dim The dimension of the input data, where the input data consists of three sets of data [passenger flow, year, types], then inp\_dim == 3\_len, bat The length of seq\_len time series, here the first 9 years are 9\*12 == 108 set, then seq\_len == 108ch\_size, inp\_dim)

Next, based on the input, we can determine the parameters of the LSTM:

rnn **=** nn**.**LSTM(inp\_dim, mid\_dim, num\_layers)

# inp\_dim is the dimension of the LSTM input tensor, we have determined this value to be 3 based on our data

# mid\_dim is the network width of the three LSTM gates (gate), and is also the dimension of the LSTM output tensor

# num\_layers is to use two LSTMs to predict the data, and then stack their outputs

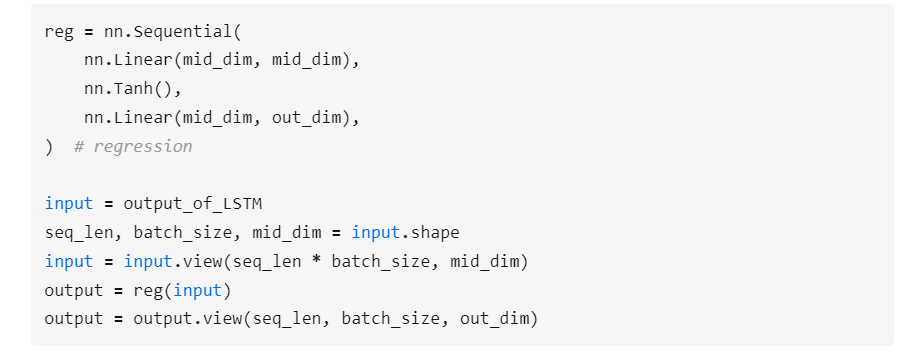
input **=** torch**.**randn(seq\_len, batch\_size, inp\_dim)

output **=** rnn(input)

**assert** output**.**size() **==** (seq\_len, batch\_size, mid\_dim)

In order to perform time series prediction, we connect two layers of fully connected layers (one layer is also acceptable) after LSTM, and change the dimension of the final output tensor at the same time. We only need to predict the value of passenger flow, so out\_dim is 1. The fully connected layer behind LSTM can also be regarded as a regression operation regression.

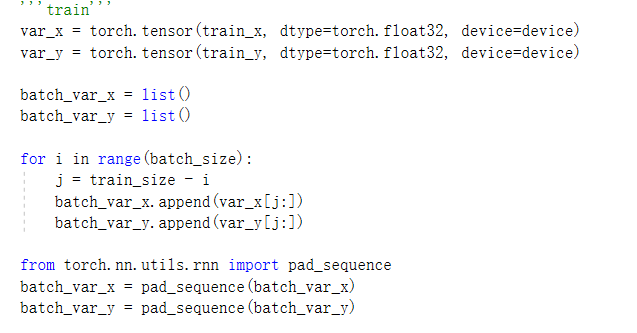
Two layers of fully connected layers are connected after LSTM, why are there two layers: Theoretically, two layers of fully connected layers that are wide enough and have at least one activation function with any kind of "squeezing" properties can fit any continuous function. The first proof of this theory was Barron et al., 1993, using UAT (Universal Approximation Theorem), pointing out that any polynomial function can be fitted in a compact domain. " In fact, for overly complex continuous functions, this "wide enough" is not easy to satisfy. And the premise of fitting the training data and making the neural network have sufficient generalization is: a good training method (such as batch training data satisfying independent and identical distribution (i.i.d.), good loss function, satisfying Lipschitz continuous etc.)



1.3 training

The length of the sequence in the same batch is different, you need to use from torch.nn.utils.rnn import pad\_sequence

We only have one set of training data, which is the previous 9 years. We can train LSTMs to predict traffic in different months in the same batch. The input of 1~t type corresponds to the passenger flow of t+1 type . Since the length of the training sequence in the same batch is not uniform, the operation of directly adding 0 at the end is not elegant, so we need the assistance of the tool pad\_sequence that comes with torch, as follows:

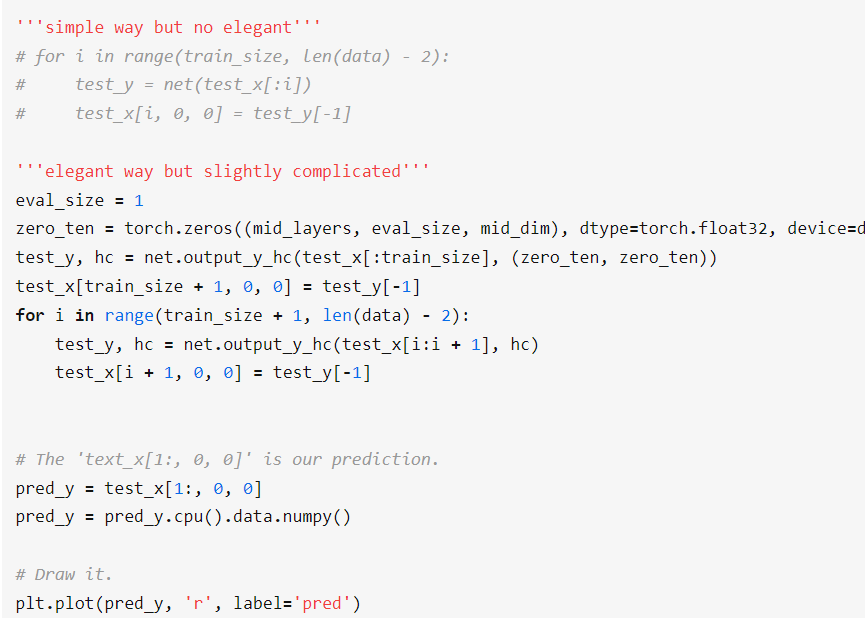


The sequences put into pad\_sequence must be placed from long to short. As the backpropagation progresses, PyTorch will gradually ignore the short sequences that complete the gradient calculation.

1.4 Evaluate

Using the data of the first 9 years as input, predict the passenger flow in the next month, and add this prediction result to the input sequence, so as to gradually predict the passenger flow in the next 3 years. It's like building a road: step on the freshly paved road and continue to pave the road.

Note that net.output\_y\_hc(self, x, hc) is used for prediction. It saves hidden state for iterative prediction, which is more elegant. The method of re-inputting the entire sequence for prediction every time is not elegant, because its computational complexity grows in O(n\*\*2) with the increase of the forecast year.



The member function output\_y\_hc defined in the REGLSTM class:We need to save the hidden state of the LSTM for resuming computation after sequence interruption. For example, I have the complete sequence seq12345: After I input seq12345 to LSTM, I can get 6 which is seq123456. I can also input seq123 first and the default hidden state hc, and get 4 and the new hc. Then I then input seq45 and hc into LSTM, and I can also get 6, namely seq123456